
ACTA CYBERNETICA

Editor-in-Chief: J. Csirik (Hungary)

Managing Editor: Z. Fülöp (Hungary)

Assistant to the Managing Editor: B. Tóth (Hungary)

Editors: L. Aceto (Denmark), M. Arató (Hungary), S. L. Bloom (USA), H. L. Bodlaender (The Netherlands), W. Brauer (Germany), L. Budach (Germany), H. Bunke (Switzerland), B. Courcelle (France), J. Demetrovics (Hungary), B. Dömölki (Hungary), J. Engelfriet (The Netherlands), Z. Ésik (Hungary), F. Gécseg (Hungary), J. Gruska (Slovakia), B. Imreh (Hungary), H. Jürgensen (Canada), A. Kelemenová (Czech Republic), L. Lovász (Hungary), G. Păun (Romania), A. Prékopa (Hungary), A. Salomaa (Finland), L. Varga (Hungary), H. Vogler (Germany), G. Wöginger (Austria)

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Contributions are accepted for review with the understanding that the same work has not been published elsewhere.

Manuscripts must be in English and can be submitted by post or by email directly to the most competent Editor in the field covered by the paper (the Editor-in-Chief and the Managing Editor should be considered as an Editor). If submitting by email, use a printable form (ps or pdf) as an attachment to the email. The email addresses of the Editors can be found in the Acta Cybernetica home page. On the first page, the *title* of the paper, the *name(s)* and *affiliation(s)*, together with the *mailing* and *electronic address(es)* of the author(s) must appear. An *abstract* summarizing the results of the paper is also required. References should be listed in alphabetical order at the end of the paper in the form which can be seen in any article already published in the journal. Manuscripts are expected to be made with a great care. If typewritten, they should be typed double-spaced on one side of each sheet. Authors are encouraged to use any available dialect of T_EX.

After acceptance, the authors will be asked to send the manuscript's source T_EX file, if any, on a diskette to the Managing Editor. Having the T_EX file of the paper can speed up the process of the publication considerably. Authors of accepted contributions may be asked to send the original drawings or computer outputs of figures appearing in the paper. In order to make a photographic reproduction possible, drawings of such figures should be on separate sheets, in India ink, and carefully lettered.

There are no page charges. Fifty reprints are supplied for each article published.

Publication information. Acta Cybernetica (ISSN 0324-721X) is published by the Department of Informatics of the University of Szeged, Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. For 2005 Numbers 1-2 of Volume 17 are scheduled. Subscription prices are available upon request from the publisher. Issues are sent normally by surface mail except to overseas countries where air delivery is ensured. Claims for missing issues are accepted within six months of our publication date. Please address all requests for subscription information to: Department of Informatics, University of Szeged, H-6701 Szeged, P.O.Box 652, Hungary. Tel.: (36)-(62)-546-396, Fax:(36)-(62)-546-397.

URL access. All these information and the contents of the last some issues are available in the Acta Cybernetica home page at <http://www.inf.u-szeged.hu/kutatas/actacybernetica/>.

EDITORIAL BOARD

Editor-in-Chief: **J. Csirik**

University of Szeged
Department of Computer Algorithms
and Artificial Intelligence
Szeged, Árpád tér 2.
H-6720 Hungary

Managing Editor: **Z. Fülöp**

University of Szeged
Department of Foundations of
Computer Science
Szeged, Árpád tér 2.
H-6720 Hungary

Assistant to the Managing Editor:

B. Tóth

University of Szeged
Research Group on
Artificial Intelligence
Szeged, Árpád tér 2.
H-6720 Hungary

Editors:

L. Aceto

Distributed Systems and Semantics Unit
Department of Computer Science
Aalborg University
Fr. Bajersvej 7E
9220 Aalborg East, Denmark

F. Gécseg

University of Szeged
Department of Computer Algorithms
and Artificial Intelligence
Szeged, Aradi vértanúk tere 1.
H-6720 Hungary

M. Arató

University of Debrecen
Department of Mathematics
Debrecen, P.O. Box 12
H-4010 Hungary

J. Gruska

Institute of Informatics/Mathematics
Slovak Academy of Science
Dúbravská 9, Bratislava 84235
Slovakia

S. L. Bloom

Stevens Institute of Technology
Department of Pure and Applied
Mathematics
Castle Point, Hoboken
New Jersey 07030, USA

B. Imreh

University of Szeged
Department of Applied Informatics
Szeged, Aradi vértanúk tere 1.
H-6720 Hungary

H. L. Bodlaender

Department of Computer Science
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

H. Jürgensen

The University of Western Ontario
Department of Computer Science
Middlesex College, London, Ontario
Canada N6A 5B7

W. Brauer
Institut für Informatik
Technische Universität München
D-80290 München
Germany

L. Budach
University of Postdam
Department of Computer Science
Am Neuen Palais 10
14415 Postdam, Germany

H. Bunke
Universität Bern
Institut für Informatik und
angewandte Mathematik
Länggass strasse 51.
CH-3012 Bern, Switzerland

B. Courcelle
Université Bordeaux-1
LaBRI, 351 Cours de la Libération
33405 TALENCE Cedex
France

J. Demetrovics
MTA SZTAKI
Budapest, Lágymányosi u. 11.
H-1111 Hungary

B. Dömölki
IQSOFT
Budapest, Teleki Blanka u. 15-17.
H-1142 Hungary

J. Engelfriet
Leiden University
LIACS
P.O. Box 9512, 2300 RA Leiden
The Netherlands

Z. Ésik
University of Szeged
Department of Foundations of
Computer Science
Szeged, Aradi vértanúk tere 1.
H-6720 Hungary

A. Kelemenová
Institute of Mathematics and
Computer Science
Silesian University at Opava
761 01 Opava, Czech Republic

L. Lovász
Eötvös Loránd University
Department of Computer Science
Budapest, Kecskeméti u. 10-12.
H-1053 Hungary

G. Păun
Institute of Mathematics
Romanian Academy
P.O.Box 1-764, RO-70700
Bucuresti, Romania

A. Prékopa
Eötvös Loránd University
Department of Operations Research
Budapest, Kecskeméti u. 10-12.
H-1053 Hungary

A. Salomaa
University of Turku
Department of Mathematics
SF-20500 Turku 50, Finland

L. Varga
Eötvös Loránd University
Department of General Computer Science
Budapest, Pázmány Péter sétány 1/c.
H-1117 Hungary

H. Vogler
Dresden University of Technology
Department of Computer Science
Foundations of Programming
D-01062 Dresden, Germany

G. Wöginger
Department of Mathematics
University of Twente
P.O. Box 217, 7500 AE Enschede
The Netherlands

On regular languages determined by nondeterministic directable automata*

Balázs Imreh[†] and Masami Ito[‡]

Abstract

It is known that the languages consisting of directing words of deterministic and nondeterministic automata are regular. Here these classes of regular languages are studied and compared. By introducing further three classes of regular languages, it is proved that the 8 classes considered form a semilattice with respect to intersection.

1 Introduction

We recall that an input word of an automaton is called *directing* or *synchronizing* if it brings the automaton from every state into the same state. An automaton is *directable* if it has a directing word. The directable automata and directing words have been studied from different points of view (see [2, 3, 5, 6, 7, 8, 10, 12, 13], for example). For nondeterministic (n.d.) automata, the directability can be defined in several ways. We study here three notions of directability which are defined in [7] as follows. An input word w of an n.d. automaton \mathcal{A} is

- (1) D1-directing if the set of states aw in which \mathcal{A} may be after reading w consists of the same single state c whatever the initial state a is;
- (2) D2-directing if the set aw is independent of the initial state a ;
- (3) D3-directing if there exists a state c included in all sets aw .

We mention that D1-directability of complete n.d. automata was already studied by Burkhard [1], where he gave an exact exponential bound for the length of minimum-length D1-directing words of complete n.d. automata. In [5], classes of languages consisting of directing words of different types of n.d. automata were studied. Here, we extend our investigations to three further classes of languages and present some of their properties. The paper is organized as follows. The next

*This work has been supported by the Japanese Ministry of Education, Mombusho International Scientific Research Program, Joint Research 10044098 and the Hungarian National Foundation for Scientific Research, Grant T037258.

[†]Dept. of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary

[‡]Dept. of Mathematics, Faculty of Science, Kyoto Sangyo University, Kyoto 603-8555, Japan

section provides general preliminaries, the formal definitions of the above language classes and some earlier results. Finally, Section 3 presents some new properties of the language families considered, in particular, it is proved that they constitute a semilattice with respect to intersection.

2 Preliminaries

Let X be a finite nonempty alphabet. As usual the set of all (finite) words over X is denoted by X^* and the empty word by ε . The length of a word w is denoted by $|w|$.

By a (deterministic) *automaton* we mean a triplet $\mathcal{A} = (A, X, \delta)$, where A is a finite nonempty set of *states*, X is the *input alphabet*, and $\delta : A \times X \rightarrow A$ is the *transition function*. This function can be extended to $A \times X^*$ in the usual way. By a *recognizer* we mean a system $\mathbf{A} = (A, X, \delta, a_0, F)$, where (A, X, δ) is an automaton, $a_0 \in A$ is the *initial state*, and $F \subseteq A$ is the set of *final states*. The *language recognized by \mathbf{A}* is the set

$$L(\mathbf{A}) = \{w \in X^* : \delta(a_0, w) \in F\}.$$

A language is called *recognizable*, or *regular*, if it is recognized by some recognizer. Sometimes, we say that the recognizer \mathbf{A} *accepts* the language $L(\mathbf{A})$.

An automaton $\mathcal{A} = (A, X, \delta)$ can also be defined as a unary algebra $\mathcal{A} = (A, X)$ for which each input letter x is realized as the unary operation $x^{\mathcal{A}} : A \rightarrow A$, $a \mapsto \delta(a, x)$. Now, nondeterministic automata can be introduced as generalized automata in which the unary operations are replaced by binary relations. Therefore, by a *nondeterministic (n.d.) automaton* we mean a system $\mathcal{A} = (A, X)$ where A is a finite nonempty set of *states*, X is the *set of the input signs (or letters)*, and each sign $x \in X$ is realized as a binary relation $x^{\mathcal{A}} \subseteq A \times A$ on A . For any $a \in A$ and $x \in X$, we define $ax^{\mathcal{A}} = \{b \in A : (a, b) \in x^{\mathcal{A}}\}$. Thus, $ax^{\mathcal{A}}$ is the set of states into which \mathcal{A} may enter from state a by reading the input letter x . For any $C \subseteq A$ and $x \in X$, we set $Cx^{\mathcal{A}} = \bigcup \{ax^{\mathcal{A}} : a \in C\}$. This transition can be extended to arbitrary $w \in X^*$ and $C \subseteq A$. $Cw^{\mathcal{A}}$ is obtained inductively by

- (1) $C\varepsilon = C$,
- (2) $Cw^{\mathcal{A}} = (Cv^{\mathcal{A}})x^{\mathcal{A}}$ for $w = vx$, $x \in X$, $w \in X^*$.

An n.d. automaton $\mathcal{A} = (A, X)$ is called *complete*, or *c.n.d. automaton*, if $ax^{\mathcal{A}} \neq \emptyset$, for all $a \in A$ and $x \in X$.

The notion of the directability of deterministic automata can be generalized to n.d. automata in several ways. The following three definitions are taken from [7]. Let $\mathcal{A} = (A, X)$ be an n.d. automaton. For any word $w \in X^*$ we consider the following three conditions:

- (D1) $(\exists c \in A)(\forall a \in A)(aw^{\mathcal{A}} = \{c\})$;
- (D2) $(\forall a, b \in A)(aw^{\mathcal{A}} = bw^{\mathcal{A}})$;
- (D3) $(\exists c \in A)(\forall a \in A)(c \in aw^{\mathcal{A}})$.

If w satisfies condition (Di) , then w is called a Di -directing word of \mathcal{A} ($i = 1, 2, 3$). For every i , $i = 1, 2, 3$, the set of Di -directing words of \mathcal{A} is denoted by $D_i(\mathcal{A})$, and \mathcal{A} is called Di -directable if $D_i(\mathcal{A}) \neq \emptyset$. It is proved (see [7]) that $D_i(\mathcal{A})$ is recognizable, for every n.d. automaton \mathcal{A} and i , $i = 1, 2, 3$. The classes of Di -directable n.d. automata and c.n.d. automata are denoted by $\text{Dir}(i)$ and $\text{CDir}(i)$, respectively.

Now, we can define the following classes of languages: For $i = 1, 2, 3$, let

$$\mathcal{L}_{\text{ND}(i)} = \{D_i(\mathcal{A}) : \mathcal{A} \in \text{Dir}(i)\} \quad \text{and} \quad \mathcal{L}_{\text{CND}(i)} = \{D_i(\mathcal{A}) : \mathcal{A} \in \text{CDir}(i)\}.$$

Finally, let \mathbf{D} denote the class of directable deterministic automata, and for any $\mathcal{A} \in \mathbf{D}$, let $D(\mathcal{A})$ be the set of directing words of \mathcal{A} . Moreover, let

$$\mathcal{L}_{\mathbf{D}} = \{D(\mathcal{A}) : \mathcal{A} \in \mathbf{D}\}.$$

Since all of the languages occurring in the definitions above are recognizable, the defined classes are subclasses of the class of the regular languages.

In what follows, we need the following definition. For any language $L \subseteq X^*$, let us denote by $P_r(L)$ the set of all prefixes of the words in L , i.e., $P_r(L) = \{u : u \in X^* \ \& \ (\exists v \in X^*)(uv \in L)\}$.

Now, we recall some results from [5] and [7] which are used in the following section.

Lemma 1 ([7]). *For any n.d. automaton $\mathcal{A} = (A, X)$, $D_2(\mathcal{A})X^* = D_2(\mathcal{A})$. If \mathcal{A} is complete, then $X^*D_1(\mathcal{A}) = D_1(\mathcal{A})$, $X^*D_2(\mathcal{A})X^* = D_2(\mathcal{A})$, and $X^*D_3(\mathcal{A})X^* = D_3(\mathcal{A})$.*

Proposition 1 ([5]). *For a language $L \subseteq X^*$, $L \in \mathcal{L}_{\mathbf{D}}$ if and only if $L \neq \emptyset$, L is regular, and $X^*LX^* = L$.*

Proposition 2 ([5]). $\mathcal{L}_{\text{CND}(2)} = \mathcal{L}_{\mathbf{D}}$, $\mathcal{L}_{\text{CND}(3)} = \mathcal{L}_{\mathbf{D}}$, $\mathcal{L}_{\text{CND}(1)} \cap \mathcal{L}_{\text{ND}(2)} = \mathcal{L}_{\mathbf{D}}$, and $\mathcal{L}_{\text{CND}(1)} \cap \mathcal{L}_{\text{ND}(3)} = \mathcal{L}_{\mathbf{D}}$.

Furthermore, we need the following proper inclusions from [5].

Remark 1 ([5]). *The following proper inclusions are valid:*

- (a) $\mathcal{L}_{\mathbf{D}} \subset \mathcal{L}_{\text{CND}(1)} \subset \mathcal{L}_{\text{ND}(1)}$,
- (b) $\mathcal{L}_{\mathbf{D}} \subset \mathcal{L}_{\text{ND}(2)}$,
- (c) $\mathcal{L}_{\mathbf{D}} \subset \mathcal{L}_{\text{ND}(3)}$.

By Proposition 2, $\mathcal{L}_{\text{CND}(3)} = \mathcal{L}_{\text{CND}(2)} = \mathcal{L}_{\mathbf{D}}$, and thus, we shall investigate the remaining 5 classes and three more defined as follows. Languages $L \subseteq X^*$ satisfying $X^*L = L$ are called *ultimate definite* (cf. [9] or [11]), and we shall consider the subclass \mathcal{U} which consists of all the regular ultimate definite languages. The second class, denoted by \mathcal{L}' , contains all the nonempty regular languages satisfying $P_r(L)LX^* = L$. Finally, we shall also consider the class $\mathcal{L}_{\text{ND}(1)} \cap \mathcal{L}_{\text{ND}(3)}$.

3 Some observations on languages of directing words of n.d. automata

First we consider the classes \mathcal{U} and $\mathcal{L}_{\text{ND}(1)}$. It is known (see [5]) that $\mathcal{L}_{\text{CND}(1)} \subset \mathcal{U}$. $\mathcal{L}_{\text{CND}(1)} \subset \mathcal{L}_{\text{ND}(1)}$ by Remark 1. The following assertion shows that $\mathcal{L}_{\text{CND}(1)}$ is the intersection of these two wider classes.

Proposition 3. $\mathcal{L}_{\text{CND}(1)} = \mathcal{L}_{\text{ND}(1)} \cap \mathcal{U}$.

Proof. As we mentioned, $\mathcal{L}_{\text{CND}(1)}$ is contained in both \mathcal{U} and $\mathcal{L}_{\text{ND}(1)}$. Therefore, it is sufficient to show that $\mathcal{L}_{\text{ND}(1)} \cap \mathcal{U} \subseteq \mathcal{L}_{\text{CND}(1)}$. For this reason, let $L \in \mathcal{L}_{\text{ND}(1)} \cap \mathcal{U}$. Then, there exists a nondeterministic D1-directable automaton $\mathcal{A} = (A, X)$ such that $L = D_1(\mathcal{A})$. We show that \mathcal{A} is a complete n.d. automaton. In order to obtain a contradiction, let us assume that there are $a' \in A$ and $x \in X$ such that $a'x^{\mathcal{A}} = \emptyset$. Let $p \in L$ be arbitrary and consider the word xp . Since $L \in \mathcal{U}$, we have $X^*L = L$, and therefore, $xp \in L$, i.e., xp is a D1-directing word. Thus, there exists a state $\bar{a} \in A$ such that $a(xp)^{\mathcal{A}} = \{\bar{a}\}$, for all $a \in A$. In particular, $a'(xp)^{\mathcal{A}} = \{\bar{a}\}$ which is a contradiction. Consequently, \mathcal{A} is a complete n.d. automaton, and thus, $L \in \mathcal{L}_{\text{CND}(1)}$. \square

Using Propositions 1 and 2, by the same argument as in the proof of Proposition 3, one can prove the following statement.

Proposition 4. $\mathcal{L}_{\text{ND}(2)} \cap \mathcal{U} = \mathcal{L}_D$ and $\mathcal{L}_{\text{ND}(3)} \cap \mathcal{U} = \mathcal{L}_D$.

By the definitions, one can easily prove the following:

Lemma 2. If $L \in \mathcal{L}_{\text{ND}(3)}$, then $P_r(L)L = L$ and $LP_r(L) = L$.

Lemma 3. If $L \in \mathcal{L}_{\text{ND}(1)}$, then $P_r(L)L = L$.

Now, we show that $\mathcal{L}_{\text{ND}(1)}$ and $\mathcal{L}_{\text{ND}(3)}$ are incomparable. To this aim, let us consider the following examples.

Example 1. Let us define the n.d. automaton $\mathcal{A} = (\{1, 2\}, \{x, y\})$ by $x^{\mathcal{A}} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$ and $y^{\mathcal{A}} = \{(1, 2), (2, 2)\}$.

Then, \mathcal{A} is D1-directable and $D_1(\mathcal{A}) = X^*y$. Now, let us suppose that $X^*y \in \mathcal{L}_{\text{ND}(3)}$. Since $y, xy \in X^*y$ and $x \in P_r(X^*y)$, by Lemma 2, we have that $yx \in X^*y$ which is a contradiction. Therefore, $\mathcal{L}_{\text{ND}(1)} \not\subseteq \mathcal{L}_{\text{ND}(3)}$.

Example 2. Let $\mathcal{A} = (\{1, 2\}, \{x, y\})$ be the n.d. automaton for which $x^{\mathcal{A}} = \{(1, 2), (2, 1), (2, 2)\}$ and $y^{\mathcal{A}} = \{(1, 1)\}$.

Now, \mathcal{A} is D3-directable and $x, x^2y \in D_3(\mathcal{A})$ while $xy \notin D_3(\mathcal{A})$. Let us suppose that $D_3(\mathcal{A}) \in \mathcal{L}_{\text{ND}(1)}$. Then, there exists an n.d. automaton $\mathcal{B} = (B, X)$ such that $D_3(\mathcal{A}) = D_1(\mathcal{B})$. In this case, x and x^2y are D1-directing words of \mathcal{B} , and thus, there are states $c, d \in B$ such that $bx^{\mathcal{B}} = \{c\}$, for all $b \in B$, in particular $cx^{\mathcal{B}} = \{c\}$, and $b(x^2y)^{\mathcal{B}} = \{d\}$ for all $b \in B$. Then, it is easy to see that $b(xy)^{\mathcal{B}} = \{d\}$, for all $b \in B$, and hence, $xy \in D_1(\mathcal{B}) = D_3(\mathcal{A})$ must hold, which is a contradiction since $xy \notin D_3(\mathcal{A})$. Consequently, $\mathcal{L}_{\text{ND}(3)} \not\subseteq \mathcal{L}_{\text{ND}(1)}$.

Regarding the class \mathcal{L}' defined by property $P_r(L)LX^* = L$, where $L \subseteq X^*$ is a nonempty regular language, the following assertion is valid.

Proposition 5. $\mathcal{L}' = \mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(3)}$.

Proof. To prove the inclusion $\mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(3)} \subseteq \mathcal{L}'$, let us suppose that $L \in \mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(3)}$. Since both classes, $\mathcal{L}_{ND(2)}$ and $\mathcal{L}_{ND(3)}$, contain nonempty regular languages (cf. [7]), L is nonempty and regular. Since $L \in \mathcal{L}_{ND(2)}$, by Lemma 1, $LX^* = L$. On the other hand, by Lemma 2, from $L \in \mathcal{L}_{ND(3)}$ it follows that $P_r(L)L = L$. Therefore, $P_r(L)LX^* = L$, and thus, $L \in \mathcal{L}'$.

In order to prove the inclusion $\mathcal{L}' \subseteq \mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(3)}$, let $L \in \mathcal{L}'$. Then, L is a nonempty regular language with $P_r(L)LX^* = L$. Since L is regular, there exists a minimal recognizer (A, X, δ, a_0, F) recognizing L . By our assumption, $LX^* = L$, and hence, by the minimality of the recognizer, we have that $F = \{f\}$ for some $f \in A$. Now, let us define the new n.d. automaton $\mathcal{B} = (B, X)$ for which $B = \{a_0q^A : q \in P_r(L)\}$ and the transitions are defined as follows. For every $b \in B$ and $x \in X$, let

$$bx^B = \begin{cases} bx^A & \text{if } bx^A \in B, \\ \emptyset & \text{otherwise.} \end{cases}$$

Now, we prove that \mathcal{B} is both D2-directable and D3-directable, moreover, $L = D_2(\mathcal{B}) = D_3(\mathcal{B})$. For this purpose, let us observe that if $p \in L$, then $a_0(qp)^B = \{f\}$, for every $q \in P_r(L)$ since $P_r(L)L = L$. Consequently, p is simultaneously a D2-directing and a D3-directing word of \mathcal{B} , moreover, $L \subseteq D_2(\mathcal{B})$ and $L \subseteq D_3(\mathcal{B})$.

To prove the inclusion $D_2(\mathcal{B}) \subseteq L$, let $p \in D_2(\mathcal{B})$ be arbitrary. Then there exists a set H of states of \mathcal{B} such that $bp^B = H$, for all $b \in B$. But, $fp^B = \{f\}$, and therefore, $H = \{f\}$, which results that $p \in L$.

For verifying $D_3(\mathcal{B}) \subseteq L$, let $p \in D_3(\mathcal{B})$ be arbitrary. Since $p \in D_3(\mathcal{B})$ and $fp^B = \{f\}$, we have $f \in bp^B$, for all $b \in B$. Then, by the definition of \mathcal{B} , $bp^B = \{f\}$, for all $b \in B$. In particular, $a_0p^B = \{f\}$, so that $a_0p^A = f$, proving $p \in L$.

Consequently, we have proved that $L \in \mathcal{L}_{ND(2)}$ and $L \in \mathcal{L}_{ND(3)}$, and therefore, $L \in \mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(3)}$. \square

Regarding the above proof, let us observe that the constructed automaton \mathcal{B} is also D1-directable, and $L = D_1(\mathcal{B})$. By this observation, one can prove the next statement in the same way as Proposition 5.

Proposition 6. $\mathcal{L}' = \mathcal{L}_{ND(2)} \cap \mathcal{L}_{ND(1)}$.

The next corollary follows from Propositions 5 and 6.

Corollary 1. $\mathcal{L}' = (\mathcal{L}_{ND(1)} \cap \mathcal{L}_{ND(3)}) \cap \mathcal{L}_{ND(2)}$.

Since $\mathcal{L}_{ND(1)}$ and $\mathcal{L}_{ND(3)}$ are incomparable with respect to set inclusion, $\mathcal{L}_{ND(1)} \cap \mathcal{L}_{ND(3)}$ is a proper subclass of both $\mathcal{L}_{ND(1)}$ and $\mathcal{L}_{ND(3)}$. Moreover, by Corollary 1, $\mathcal{L}' \subseteq \mathcal{L}_{ND(1)} \cap \mathcal{L}_{ND(3)}$ and $\mathcal{L}' \subseteq \mathcal{L}_{ND(2)}$. Both inclusions are proper. To verify this observation, let us consider the following examples.

Example 3. Let the n.d. automaton $\mathcal{A} = (\{1, 2\}, X)$ be defined by $X = \{x, y\}$, $x^{\mathcal{A}} = \{(2, 1), (2, 2)\}$, and $y^{\mathcal{A}} = \{(1, 1), (2, 1)\}$.

Then, y is a D1- and D3-directing word, and $L = y\{y\}^* = D_1(\mathcal{A}) = D_3(\mathcal{A})$. Now, if $L \in \mathcal{L}'$, then $P_r(L)LX^* = L$ must hold, which is a contradiction since $y^k x \notin L$, for every integer $k \geq 1$. Therefore, $\mathcal{L}' \subset \mathcal{L}_{ND(1)} \cap \mathcal{L}_{ND(3)}$.

Example 4. Let the n.d. automaton $\mathcal{A} = (\{1, 2\}, X)$ be defined by $X = \{x, y\}$, $x^{\mathcal{A}} = \{(1, 2), (2, 2)\}$, and $y^{\mathcal{A}} = \{(2, 1)\}$.

Then, \mathcal{A} is D2-directable and $D_2(\mathcal{A}) = xX^* \cup X^*y^2X^*$. Now, if $D_2(\mathcal{A}) \in \mathcal{L}'$, then since $y \in P_r(D_2(\mathcal{A}))$ and $x \in D_2(\mathcal{A})$, $yx \in D_2(\mathcal{A})$ must hold, which is a contradiction. Consequently, $\mathcal{L}' \subset \mathcal{L}_{CND(2)}$.

By the definition of \mathcal{L}' and Proposition 1, we obviously have that $\mathcal{L}_D \subseteq \mathcal{L}'$. For proving that this inclusion is proper, let us consider the following example.

Example 5. Let $\mathcal{A} = (\{1, 2\}, X)$, where $X = \{x, y\}$, $x^{\mathcal{A}} = \{(2, 2)\}$, and $y^{\mathcal{A}} = \{(1, 2), (2, 2)\}$.

Then, $D_1(\mathcal{A}) = D_2(\mathcal{A}) = D_3(\mathcal{A}) = yX^*$. By Proposition 5, $yX^* \in \mathcal{L}'$. Let us suppose now that $yX^* \in \mathcal{L}_D$. Then, by Proposition 1, $xy \in yX^*$ must hold, which is a contradiction. Therefore, $yX^* \notin \mathcal{L}_D$, and thus, $\mathcal{L}_D \subset \mathcal{L}'$.

Summarizing, we obtain the following result.

Theorem 1. *If $|X| \geq 2$, then the 8 classes under consideration constitute a semilattice with respect to intersection.*

The semilattice of these classes is depicted in Figure 1.

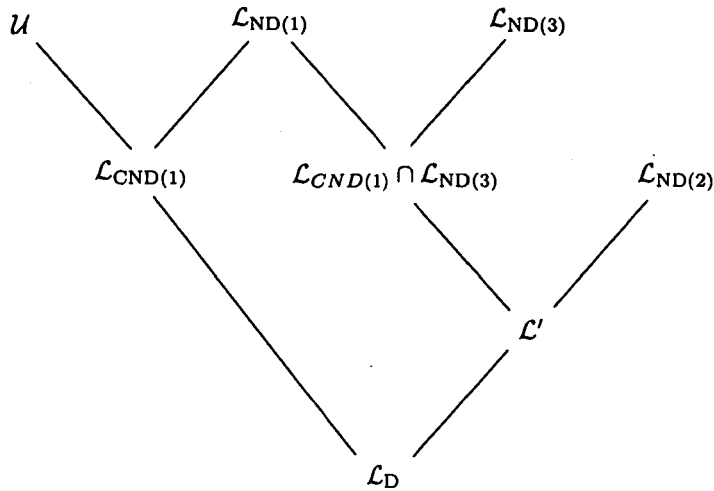


Figure 1: Semilattice of the classes considered.

Let $\mathcal{A} = (A, X)$ be an n.d. automaton and $x \in X$. Then, x is called a *complete input sign* if $ax^A \neq \emptyset$, for all $a \in A$.

The following statement shows that the languages belonging to $\mathcal{L}_{ND(2)}$ can be decomposed into a particular form.

Proposition 7. *If $L \in \mathcal{L}_{ND(2)}$, then L is a disjoint union of regular languages L_1 and L_2 where at least one of L_1 and L_2 is nonempty, furthermore,*

- (1) $L_1 \in \mathcal{L}_D$ or $L_1 = \emptyset$,
and
- (2) $L_2 = P_r(L_2)L_2Y^*$ and $L_2 = Y^*L_2Y^*$, where $Y \subseteq X$ denotes the set of complete input symbols of \mathcal{A} , or $L_2 = \emptyset$.

Proof. Let $L \in \mathcal{L}_{ND(2)}$ be arbitrary. Then, there exists a D2-directable n.d. automaton $\mathcal{A} = (A, X)$ such that $L = D_2(\mathcal{A})$, i.e., L consists of the D2-directing words of \mathcal{A} . Let us classify now the D2-directing words of \mathcal{A} as follows. Let

$$\begin{aligned} L_1 &= \{p : p \in L \text{ \& } ap^A = \emptyset, \text{ for all } a \in A\}, \\ L_2 &= \{p : p \in L \text{ \& } ap^A \neq \emptyset, \text{ for some } a \in A\}. \end{aligned}$$

Obviously, $L_1 \cap L_2 = \emptyset$ and $L_1 \cup L_2 = L$, furthermore, one of the languages L_1 and L_2 is nonempty.

Let us suppose that $L_1 \neq \emptyset$. It is easy to see that L_1 is regular. Now, if $p \in L_1$, then $ap^A = \emptyset$, for all $a \in A$. Thus also $a(qpr)^A = \emptyset$, for all $q, r \in X^*$ and $a \in A$. Therefore, $X^*L_1X^* = L_1$, and by Proposition 1, we obtain that $L_1 \in \mathcal{L}_D$ if $L_1 \neq \emptyset$.

The regularity of L_2 can be concluded by the fact that $L_2 = L \setminus L_1$. Let us observe that $Y = \emptyset$ implies $L_2 = \emptyset$.

Now, let us suppose that $L_2 \neq \emptyset$ and let $p \in L_2$ and $q \in P_r(L_2)$. Then, there exists an $r \in X^*$ with $qr \in L_2$. Since $qr \in L_2$, $a(qr)^A \neq \emptyset$, for all $a \in A$. Therefore, $aq^A = A_a \neq \emptyset$, for all $a \in A$. Furthermore, since $p \in L_2$, we have that there exists a nonempty set H of states such that $A'p^A = H$, for every nonempty subset A' of A . In particular, $A_ap^A = H$, for all $a \in A$. Consequently, $a(qp)^A = (aq^A)p^A = A_ap^A = H$, for all $a \in A$, and hence, $qp \in L_2$. On the other hand, since Y is the set of complete input signs, $L_2Y^* = L_2$.

To prove the second equality, let $q \in Y^*$ and $p \in L_2$ be arbitrary words. From $p \in L_2$ it follows again that there exists a nonempty set H of states such that $A'p^A = H$, for all nonempty subsets A' of A . On the other hand, since $q \in Y^*$, $aq^A \neq \emptyset$, for all $a \in A$. Consequently, $H = aq^Ap^A = a(qp)^A$, for all $a \in A$, and thus, $Y^*L_2 = L_2$. The validity of the equality $L_2Y^* = L_2$ is obvious, and hence, $Y^*L_2Y^* = L_2$. \square

Now, we study the representation of the languages of $\mathcal{L}_{ND(2)}$ which have the form $L = MX^*$, where M is a regular prefix code. For this reason, we recall some notions.

Let $\emptyset \neq M \subseteq X^+$. Then, M is said to be a *prefix code* over X if $M \cap MX^+ = \emptyset$. A prefix code $M \subseteq X^+$ is said to be *maximal* if, for any $u \in X^*$, there exists $v \in X^*$ such that $uv \in MX^+$. Finally, a prefix code M is called *regular* if M is a regular language. Note that any $L \in \mathcal{L}_{\text{ND}(2)}$ can be represented as $L = MX^*$ such that $M = L \setminus LX^+$ and M is a prefix code because $LX^* = L$.

Proposition 8. *Let $M \subseteq X^+$ be a regular prefix code that is not maximal. Let $L = MX^*$. Then, $L \in \mathcal{L}_{\text{ND}(2)}$ if and only if $P_r(M)M \subseteq L$.*

Proof. To prove the necessity, let us assume $L \in \mathcal{L}_{\text{ND}(2)}$. Then, there exists an n.d. automaton $\mathcal{A} = (A, X)$ such that $L = D_2(\mathcal{A})$. Let $u \in P_r(M)$ and $w \in M$. Since $u \in P_r(M)$, there exists $v \in X^*$ such that $uv \in M \subseteq L$. Hence, for any $a, b \in A$, $a(uv)^A = b(uv)^A$. Suppose $a(uv)^A = \emptyset$ for any $a \in A$. Then, for any $a \in A$ and $z \in X^*$, $a(z(uv))^A = \emptyset$. This yields that $zuv \in L$, for all $z \in X^*$, and hence, M is a maximal prefix code, which is a contradiction. Therefore, $a(uv)^A \neq \emptyset$, and thus, $au^A \neq \emptyset$, for all $a \in A$. Consequently, $a(uw)^A = b(uw)^A$ for any $a, b \in A$ since $w \in M \subseteq L$. Thus, $uw \in L$.

In order to prove the sufficiency, let $\mathbf{A}' = (A, X, a_0, \delta, F)$ be the minimal recognizer (deterministic but not necessarily complete) accepting L . Notice that \mathbf{A}' is a trim (i.e. accessible and coaccessible, see [4]) and $F = \{f\}$, since M is a prefix code and $L = MX^*$. Consider the n.d. automaton $\mathcal{A} = (A, X)$. Note that $fx^A = \{f\}$ for any $x \in X$. Let $a \in A$ and $w \in L$. Since \mathbf{A}' is trim, there exist $u, v \in X^*$ such that $\{a\} = a_0u^A$ and $a_0(uv)^A = \{f\}$, i.e., $uv \in L$. Consequently, $u \in P_r(M)$ or $u \in MX^*$. If $u \in P_r(M)$, then $uw \in P_r(M)MX^* \subseteq LX^* = L$. If $u \in MX^*$, then $uw \in MX^*X^* = MX^* = L$. Hence, $aw^A = \{f\}$, for all $a \in A$. This means that $w \in D_2(\mathcal{A})$. Now, let $w \notin L$. In this case, $fw^A = \{f\}$ but $a_0w^A \neq \{f\}$. This means that $w \notin D_2(\mathcal{A})$. Consequently, $L = D_2(\mathcal{A})$. This completes the proof of the proposition. \square

The above proposition does not always hold for a regular maximal prefix code.

Example 6. Let $X = \{x, y\}$ and let $A = \{1, 2\}$. Moreover, let $\mathcal{A} = (A, X)$ be the following n.d. automaton: $x^A = \{(1, 2), (2, 2)\}$, $y^A = \{(1, 2)\}$.

Then, $L = D_2(\mathcal{A}) = (x \cup yx^*y)X^* \in \mathcal{L}_{\text{ND}(2)}$. Let $M = L \setminus LX^+$. Then, $P_r(M)M \subseteq L$ does not hold since $y \in P_r(M)$, $x \in M$ but $yx \notin L = MX^*$.

However, for the class of finite maximal prefix codes, we have the following:

Proposition 9. *Let $\emptyset \neq M \subseteq X^+$ be a finite maximal prefix code. Let $L = MX^*$. Then, $L \in \mathcal{L}_{\text{ND}(2)}$ if and only if $P_r(M)M \subseteq L$.*

Proof. The sufficiency can be proved in the same way as in the proof of the previous proposition. To prove the necessity, let us assume that $L = MX^* \in \mathcal{L}_{\text{ND}(2)}$. Let $\mathcal{A} = (A, X)$ be an n.d. automaton such that $L = D_2(\mathcal{A})$. Let $u \in P_r(M)$ and $w \in M$. Since M is a finite maximal prefix code, $uw^i \in MX^*$ for some $i, i \geq 1$. There are two cases. First, assume $a(uw^i)^A \neq \emptyset$ for any $a \in A$. In this case, $au^A \neq \emptyset$ for any $a \in A$. Since $w \in M \subseteq L$, $(au^A)w^A = (bu^A)w^A$ for any

$a, b \in A$. Thus, $a(uw)^A = b(uw)^A$ for any $a, b \in A$. This means that $uw \in L$. Now, assume $a(uw^i)^A = \emptyset$ for any $a \in A$. Suppose that there exists $a \in A$ such that $a(uw)^A \neq \emptyset$. In this case, there exists a nonempty subset H of A such that $(au^A)w^A = H \neq \emptyset$. Thus, $Hw^A = H$ holds because $w \in L$. This implies that $a(uw^i)^A = (a(uw^{i-1})^A)w^A = H \neq \emptyset$, a contradiction. Consequently, $a(uw)^A = \emptyset$ for any $a \in A$, and hence $uw \in L$. In either case, $uw \in L$, completing the proof of the proposition. \square

Example 7. Let $X = \{x, y\}$ and let $M = \{x, yxx, yxy, yy\}$. Then, M is a finite maximal prefix code. Take $y \in P_r(M)$ and $x \in M$. Then, $yx \notin MX^*$. Therefore, $MX^* \notin \mathcal{L}_{ND(2)}$.

References

- [1] H.V. Burkhard, Zum Längenproblem homogener experimente an determinierten und nicht-deterministischen automaten, *Elektronische Informationsverarbeitung und Kybernetik*, *EIK* **12** (1976), 301-306.
- [2] J. Černý, Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny Časopis SAV* **14** (1964), 208-215.
- [3] J. Černý, A. Pirická & B. Rosenauerová, On directable automata, *Kybernetika (Praha)* **7** (1971), 289-297.
- [4] S. Eilenberg, *Automata, Languages and Machines* Vol. A, Academic Press, New York and London, 1974.
- [5] B. Imreh & M. Ito, On some special classes of regular languages, in *Jewels are Forever* (eds.: J. Karhumäki, H. Maurer, G. Paun and G. Rozenberg), Springer-Verlag, Berlin-Heidelberg-New York-Tokyo, 1999, 25-34.
- [6] B. Imreh & M. Steinby, Some remarks on directable automata, *Acta Cybernetica* **12** (1995), 23-35.
- [7] B. Imreh & M. Steinby, Directable nondeterministic automata, *Acta Cybernetica* **14** (1999), 105-115.
- [8] M. Ito & J. Duske, On cofinal and definite automata. *Acta Cybernetica* **6** (1983), 181-189.
- [9] A. Paz & B. Peleg, Ultimate-definite and symmetric definite events and automata, *J. ACM* **12** (1965), 399-410.
- [10] J.-E. Pin, Sur les mots synchronisants dans un automata fini, *Elektronische Informationsverarbeitung und Kybernetik*, *EIK* **14** (1978), 297-303.
- [11] R.G. Reynolds & W.F. Cutlip, Synchronization and general repetitive machines, with applications to ultimate definite automata, *J. ACM* **16** (1969), 226-234.

- [12] I. Rystsov, Reset words for commutative and solvable automata, *Theoretical Computer Science* **172** (1997), 273-279.
- [13] P.H. Starke, *Abstrakte Automaten*, VEB Deutscher Verlag der Wissenschaften, Berlin 1969.

Received December, 2002

Generation of Sentences with Their Parses: the Case of Propagating Scattered Context Grammars

Alexander Meduna* and Jiří Techet*

Abstract

Propagating scattered context grammars are used to generate their sentences together with their parses—that is, the sequences of labels denoting productions whose use lead to the generation of the corresponding sentences. It is proved that for every recursively enumerable language L , there exists a propagating scattered context grammar whose language consists of L 's sentences followed by their parses.

Keywords: parsing, propagating scattered context grammars

1 Introduction

Parallel parsing represents a vivid investigation area concerning compilers today (see [1, 2, 9, 10, 16]). As parsing is almost always based on suitable grammatical models, parallel grammars are important to this area. Since scattered context grammars generate their languages in a parallel way, their use related to parsing surely deserves our attention.

In this paper, we use the propagating scattered context grammars, which contain no erasing productions, to generate their language's sentences together with their parses—that is, the sequences of labels denoting productions whose use lead to the generation of the corresponding sentences (in the literature, derivations words and Szilard words are synonymous with parses). We demonstrate that for every recursively enumerable language L , there exists a propagating scattered context grammar whose language consists of L 's sentences followed by their parses. That is, if we eliminate all the suffixes representing the parses, we obtain precisely L . This characterization of recursively enumerable languages is of some interest because it is based on propagating scattered context grammars whose languages are included in the family of context-sensitive languages, which is properly contained in the family of recursively enumerable languages. Simply stated, in this paper, we use the propagating scattered context grammars in such a way that this use provides us with the parses corresponding to the generated sentences and, in addition, increases the generative power of these grammars.

*Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 61266, Czech Republic

2 Preliminaries

We assume that the reader is familiar with the language theory (see [6, 11, 12, 13]). For an alphabet V , $\text{card}(V)$ denotes the cardinality of V . V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ϵ . Set $V^+ = V^* - \{\epsilon\}$. For $w \in V^*$, $|w|$ and $\text{rev}(w)$ denote the length of w and the reversal of w , respectively. For $U \subseteq V$, $\text{occur}(w, U)$ denotes the number of occurrences of symbols from U in w . For $L \subseteq V^*$, $\text{alph}(L)$ denotes the set of symbols appearing in a word of L . Let L_1, L_2 be two languages. The *right quotient* of L_1 with respect to L_2 , denoted by L_1/L_2 , is defined as $L_1/L_2 = \{y \mid yx \in L_1, \text{ for some } x \in L_2, y \in \text{alph}(L_1)^*\}$. The *left quotient* of L_1 with respect to L_2 , denoted by $L_2 \backslash L_1$, is defined as $L_2 \backslash L_1 = \{y \mid xy \in L_1, \text{ for some } x \in L_2, y \in \text{alph}(L_1)^*\}$.

A *scattered context grammar* (see [3, 4, 5, 7, 8, 14, 15] and pages 259–260 in [13]), a *SCG* for short, is a quadruple, $G = (V, P, S, T)$, where V is an alphabet, $T \subseteq V$, $S \in V - T$, and P is a finite set of productions such that each production has the form $(A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n)$, for some $n \geq 1$, where $A_i \in V - T$, $x_i \in V^*$, for $1 \leq i \leq n$. If every $(A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n) \in P$ satisfies $x_i \in V^+$ for all $1 \leq i \leq n$, G is a *propagating scattered context grammar*, a *PSCG* for short. If $(A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n) \in P$, $u = u_1 A_1 u_2 \dots u_n A_n u_{n+1}$, and $v = u_1 x_1 u_2 \dots u_n x_n u_{n+1}$, where $u_i \in V^*$, $1 \leq i \leq n$, then $u \Rightarrow v [(A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n)]$ in G or, simply, $u \Rightarrow v$. Let \Rightarrow^+ and \Rightarrow^* denote the transitive closure of \Rightarrow and the transitive-reflexive closure of \Rightarrow , respectively. The *language of G* is denoted by $L(G)$ and defined as $L(G) = \{x \mid x \in T^*, S \Rightarrow^* x\}$.

3 Definitions and examples

Throughout this paper, we assume that for every SCG $G = (V, P, S, T)$, there is a set of production labels denoted by $\text{lab}(G)$ such that $\text{card}(\text{lab}(G)) = \text{card}(P)$; as usual, $\text{lab}(G)^*$ denotes the set of all strings over $\text{lab}(G)$. Let us label each production in P uniquely with a label from $\text{lab}(G)$ so that this labeling represents a bijection from $\text{lab}(G)$ to P . To express that $p \in \text{lab}(G)$ labels a production $(A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n)$, we write $p : (A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n)$. For every $p : (A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n) \in P$, $\text{lhs}(p)$ and $\text{rhs}(p)$ denote $A_1 A_2 \dots A_n$ and $x_1 x_2 \dots x_n$, respectively. Furthermore, $\text{ipos}(p, j)$ and $\text{rpos}(p, j)$ denote A_j and x_j , respectively. To express that G makes $x \Rightarrow^* y$ by using a sequence of productions labeled by p_1, p_2, \dots, p_n , we write $x \Rightarrow^* y [\rho]$, where $x, y \in V^*$, $\rho = p_1 \dots p_n \in \text{lab}(G)^*$. Let $S \Rightarrow^* x [\rho]$ in G , where $x \in T^*$ and $\rho \in \text{lab}(G)^*$; then, x is a *sentence generated by G according to parse ρ* . Let $G = (V, P, S, T)$ be a SCG with $\text{lab}(G) \subseteq T$. G is a *proper generator of its sentences with their parses* if $L(G) = \{x \mid x = y\rho, y \in (T - \text{lab}(G))^*, \rho \in \text{lab}(G)^*, S \Rightarrow^* x [\rho]\}$.

Next, we illustrate these definitions by three SCGs, each of which has its set of production labels equal to $\{1, 2, 3, 4\}$. First, consider SCG $G_1 = (\{S, A, B, C, a, b, c\}, P_1, S, \{a, b, c\})$ with P_1 containing $1 : (S) \rightarrow (e)$, $2 : (S) \rightarrow$

(ABC) , $3 : (A, B, C) \rightarrow (aA, bB, cC)$, $4 : (A, B, C) \rightarrow (a, b, c)$. As $\{1, 2, 3, 4\} \not\subseteq \{a, b, c\}$, G_1 is no proper generator of its sentences with their parses. Second, consider $G_2 = (\{S, A, B, C, a, b, c, 1, 2, 3, 4\}, P_2, S, \{a, b, c, 1, 2, 3, 4\})$ with P_2 containing $1 : (S) \rightarrow (1)$, $2 : (S) \rightarrow (ABC2)$, $3 : (A, B, C) \rightarrow (aA, bB, cC3)$, $4 : (A, B, C) \rightarrow (a, b, c4)$. Notice that $\{1, 2, 3, 4\} \subseteq \{a, b, c, 1, 2, 3, 4\}$. However, $L(G_2) = \{a^n b^n c^n \text{rev}(\rho) \mid n \geq 0, S \Rightarrow^* a^n b^n c^n \text{rev}(\rho) [\rho]\} \neq \{a^n b^n c^n \rho \mid n \geq 0, S \Rightarrow^* a^n b^n c^n \rho [\rho]\}$, so G_2 is no proper generator of its sentences with their parses either. Third, consider $G_3 = (\{S, A, B, C, a, b, c, 1, 2, 3, 4\}, P_3, S, \{a, b, c, 1, 2, 3, 4\})$ with P_3 containing $1 : (S) \rightarrow (1)$, $2 : (S) \rightarrow (ABC2\$)$, $3 : (A, B, C, \$) \rightarrow (aA, bB, cC, 3\$)$, $4 : (A, B, C, \$) \rightarrow (a, b, c, 4)$. Observe that $L(G_3) = \{a^n b^n c^n \rho \mid n \geq 0, S \Rightarrow^* a^n b^n c^n \rho [\rho]\}$, so G_3 is a proper generator of its sentences with their parses.

4 Results

Next, we demonstrate that for every recursively enumerable language L , there is a PSCG $G = (V, P, S, T)$, which represents a proper generator of its sentences with their parses so that L results from $L(G)$ by eliminating all production labels in $L(G)$. To express this property formally, we introduce the weak identity π from V^* to $(V - \text{lab}(G))^*$ defined as $\pi(a) = a$ for every $a \in (V - \text{lab}(G))$ and $\pi(p) = \epsilon$ for every $p \in \text{lab}(G)$ and use π in the next main theorem of this paper.

Theorem 1. *For every recursively enumerable language L , there exists a PSCG G such that G is a proper generator of its sentences with their parses and $L = \pi(L(G))$.*

Proof. Let L be a recursively enumerable language. Then, there is a SCG $\bar{G} = (\bar{V}, \bar{P}, \bar{S}, \bar{T})$ such that $L = L(\bar{G})$ (see [7]). Set $\Phi = \{\langle a \rangle \mid a \in \bar{T}\}$. Define the homomorphism γ from \bar{V} to $(\Phi \cup (\bar{V} - \bar{T}) \cup \{Y\})^+$ as $\gamma(a) = \langle a \rangle$ for all $a \in \bar{T}$ and $\gamma(A) = A$ for all $A \in \bar{V} - \bar{T}$. Extend the domain of γ to \bar{V}^+ in the standard manner; non-standardly, however, define $\gamma(\epsilon) = Y$ rather than $\gamma(\epsilon) = \epsilon$. (Let us note that at this point γ does not, strictly speaking, represent a morphism on \bar{V}^* .) Next, we introduce a PSCG $G = (V, P, S, T)$ such that G is a proper generator of its sentences with their parses and $L(\bar{G}) = \pi(L(G))$. Finally, set $\Gamma = \{\$1, \$2, \$3\}$. Define the PSCG

$$G = (\{S, X, Y, Z\} \cup \bar{V} \cup \text{lab}(G) \cup \Phi \cup \Gamma, P, S, \bar{T} \cup \text{lab}(G))$$

with $\text{lab}(G) = \{\langle 0 \rangle, \langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle\} \cup \Xi_1 \cup \Xi_2 \cup \Xi_3$, where $\Xi_1 = \{\langle p1 \rangle \mid p \in \text{lab}(\bar{G})\}$, $\Xi_2 = \{\langle a2 \rangle \mid a \in \bar{T}\}$, $\Xi_3 = \{\langle a3 \rangle \mid a \in \bar{T}\}$; without any loss of generality, assume $\text{lab}(G) \cap \text{alph}(L) = \emptyset$. P is constructed as follows:

1. Add

$$\begin{aligned} [1] : (S) &\rightarrow (X[1]\$1Z\bar{S}) \text{ to } P; \\ [1_\epsilon] : (S) &\rightarrow ([1_\epsilon]\$1\bar{S}) \text{ to } P; \end{aligned}$$

2. For every $p : (A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n) \in \bar{P}$ add
 $[p1] : (\$1, A_1, \dots, A_n) \rightarrow ([p1]\$1, \gamma(x_1), \dots, \gamma(x_n))$ to P ;
 in addition, add
 $[2] : (\$1) \rightarrow ([2]\$2)$ to P ;
 $[2_\epsilon] : (\$1) \rightarrow ([2_\epsilon]\$3)$ to P ;
3. For every $a \in \bar{T}$, add
 $[a2] : (X, \$2, Z, \langle a \rangle) \rightarrow (aX, [a2]\$2, Y, Z)$ to P ;
 $[a3] : (X, \$2, Z, \langle a \rangle) \rightarrow (a, [a3]\$3, Y, Y)$ to P ;
4. Add $[3] : (\$3, Y) \rightarrow ([3]\$3)$ to P ;
5. Add $[4] : (\$3) \rightarrow ([4])$ to P .

Basic Idea:

First, we explain how G makes the generation of a nonempty sentence followed by its parse; then, we explain the generation of the empty sentence followed by its parse.

G makes the generation of $a_1 a_2 \dots a_n \rho$, where $n \geq 1$, each $a_i \in \bar{T}$ and ρ is the corresponding parse, by productions introduced in steps 1 through 5 in this order. After starting this generation by using the production from 1, it applies productions introduced in 2, which simulate the applications of productions from \bar{P} . More precisely, it simulates the use of $p : (A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n) \in \bar{P}$ by using $[p1] : (\$1, A_1, \dots, A_n) \rightarrow ([p1]\$1, \gamma(x_1), \dots, \gamma(x_n)) \in P$ so that it places its own label, $[p1]$, right behind the previously generated production labels; this substring of labels occurs between the leftmost symbol, X , and $\$1$, in the sentential form. Otherwise, $[p1] : (\$1, A_1, \dots, A_n) \rightarrow ([p1]\$1, \gamma(x_1), \dots, \gamma(x_n))$ is analogical to $p : (A_1, \dots, A_n) \rightarrow (x_1, \dots, x_n)$ except that (i) the former has the fill-in symbol Y where the latter has ϵ and (ii) the former has $\langle a_i \rangle$ where the latter has terminal a_i . After using productions introduced in 2, G has its current sentential form of the form $X \tau \$2 Z u_0 \langle a_1 \rangle u_1 \langle a_2 \rangle u_2 \dots u_{n-1} \langle a_n \rangle u_n$, where τ is a prefix of ρ and $u_i \in \{Y\}^*$. By using productions from 3, it places $a_1 \dots a_n$ at the beginning of the sentential form while replacing each $\langle a_i \rangle$ with Y and generating the production labels. By using productions labeled $[3]$ (see step 4), G replaces each Y with $[3]$ while shifting $\$3$ to the right. Finally, the application of the production labeled with $[4]$ completes the generation of $a_1 a_2 \dots a_n \rho$ (see step 5). Finally, let us explain how G makes the generation of the empty sentence ϵ followed by its parse. By use of productions labeled with $[1_\epsilon]$ and $[2_\epsilon]$ instead of $[1]$ and $[2]$, respectively, the process of placing terminal symbols at the beginning of the sentential form (by productions from step 3) is skipped; otherwise, the derivation proceeds as above.

Rigorous proof (Sketch):

Claim 1. G generates every $w \in L(G) - \text{lab}(G)^+$ in the following way

$$\begin{array}{lll}
 S & \Rightarrow & X[1]\$_1 Z \bar{S} \quad [[1]] \\
 & \Rightarrow^+ & x \quad [\rho] \\
 & \Rightarrow & y \quad [[2]] \\
 & \Rightarrow^* & z \quad [\sigma] \\
 & \Rightarrow & u \quad [[a3]] \\
 & \Rightarrow^+ & v \quad [\tau] \\
 & \Rightarrow & w \quad [[4]]
 \end{array} \tag{1}$$

where $[a3] \in \Xi_3$, ρ , σ and τ are sequences consisting from Ξ_1 , Ξ_2 and $\{[3]\}$, respectively.

Proof. First, let us make these four observations:

1. Since the only productions with S on its left-hand side are productions introduced in step 1 of the construction, $S \Rightarrow^+ w$ surely starts with a step made by one of these productions. Notice that $\text{alph}(\{w\}) \cap \bar{T} \neq \emptyset$ and only productions labeled with $p \in \Xi_2 \cup \Xi_3$ satisfy $a \in \text{alph}(\{\text{rhs}(p)\})$, $a \in \bar{T}$. As $X = \text{pos}(p, 1)$, $a \in \text{alph}(\{\text{pos}(p, 1)\})$, and only production labeled with $p \in [1]$ satisfies $X \in \text{alph}(\{\text{rhs}(p)\})$, the derivation starts with a step made by this production. This derivation ends by applying production labeled with $[4]$ because it is the only production with its right-hand side over T^* . Thus, $S \Rightarrow^+ w$ can be expressed as

$$\begin{array}{lll}
 S & \Rightarrow & X[1]\$_1 Z \bar{S} \quad [[1]] \\
 & \Rightarrow^+ & v \\
 & \Rightarrow & w \quad [[4]]
 \end{array}$$

2. Let p be the label of any production introduced in steps 2 through 4 of the construction; then, $\text{occur}(\text{lhs}(p), \Gamma) = \text{occur}(\text{rhs}(p), \Gamma) = 1$. In greater detail, for every $[p1] \in \Xi_1$, $[a2] \in \Xi_2$, $[a3] \in \Xi_3$, productions introduced in step 2 satisfy $\text{occur}(\text{lhs}([p1]), \{\$1\}) = \text{occur}(\text{rhs}([p1]), \{\$1\}) = 1$, $\text{occur}(\text{lhs}([2]), \{\$1\}) = 1$, $\text{occur}(\text{rhs}([2]), \{\$2\}) = 1$, $\text{occur}(\text{lhs}([2_e]), \{\$1\}) = 1$, $\text{occur}(\text{rhs}([2_e]), \{\$3\}) = 1$. Similarly, productions introduced in step 3 satisfy $\text{occur}(\text{lhs}([a2]), \{\$2\}) = \text{occur}(\text{rhs}([a2]), \{\$2\}) = 1$, $\text{occur}(\text{lhs}([a3]), \{\$2\}) = 1$, $\text{occur}(\text{rhs}([a3]), \{\$3\}) = 1$. Finally, production introduced in step 4 satisfies $\text{occur}(\text{lhs}([3]), \{\$3\}) = \text{occur}(\text{rhs}([3]), \{\$3\}) = 1$.
3. Because $X \in \text{alph}(\{x\})$ and only productions labeled with $p \in \Xi_3$ satisfy $X \in \text{alph}(\{\text{lhs}(p)\})$ and $X \notin \text{alph}(\{\text{rhs}(p)\})$, production labeled with $[2_e]$ cannot be used.
4. Let p be the label of any production introduced in steps 1 through 5; then, $\text{alph}(\{\text{rhs}(p)\}) \cap \text{lab}(G) = \{p\}$ and $\text{occur}(\text{rhs}(p), \{p\}) = 1$.

Based on these observations, notice that G generates every $w \in L(G) - \{[0]\}$ in the way described in the formulation of Claim 1. \square

Claim 2. Consider derivation (1). In its beginning

$$\begin{array}{lll} S & \Rightarrow & X[1]\$_1Z\bar{S} \quad [[1]] \\ & \Rightarrow^+ & x \quad [\rho] \\ & \Rightarrow & y \quad [[2]] \end{array}$$

every sentential form s in $X[1]\$_1Z\bar{S} \Rightarrow^+ x$ satisfies $s \in \{X\}lab(G)^+\{\$1\}\{Z\}(\Phi \cup (\bar{V} - \bar{T}) \cup \{Y\})^+$ and $y \in \{X\}lab(G)^+\{\$2\}\{Z\}(\Phi \cup \{Y\})^+$.

Proof. By the definition of homomorphism γ , productions labeled with $[p1]$ rewrite symbols over $\Phi \cup (\bar{V} - \bar{T}) \cup \{Y\}$ and change $\$1$ to $[p1]\$1$. Since $\bar{V} \cap \{X, \$1, Z\} = \emptyset$, every sentential form s in $X[1]\$_1Z\bar{S} \Rightarrow^+ x$ satisfies $s \in \{X\}lab(G)^+\{\$1\}\{Z\}(\Phi \cup (\bar{V} - \bar{T}) \cup \{Y\})^+$. Only Ξ_1 contains production labels p satisfying $alph(\{lhs(p)\}) \cap (\bar{V} - \bar{T}) \neq \emptyset$. Therefore, to generate $w \in T^*$, productions labeled with $[p1]$ have to be applied until $s \in \{X\}lab(G)^+\{\$1\}\{Z\}(\Phi \cup \{Y\})^+$. Finally, a production labeled with $[2]$ is used, so $y \in \{X\}lab(G)^+\{\$2\}\{Z\}(\Phi \cup \{Y\})^+$ and the claim holds. \square

Claim 3. In

$$\begin{array}{lll} y & \Rightarrow^* & z \quad [[\sigma]] \\ & \Rightarrow & u \quad [[a3]] \end{array}$$

of derivation (1), every sentential form o in $y \Rightarrow^* z$ can be expressed as $o \in \bar{T}^*\{X\}lab(G)^+\{\$2\}\{Y\}^*\{Z\}(\Phi \cup \{Y\})^+$ and $u \in \bar{T}^+lab(G)^+\{\$3\}\{Y\}^+$. In greater detail,

$$\begin{array}{lll} & X[p_1] \dots [p_n]\$2ZY^{i_0}\langle b_1 \rangle Y^{i_1}\langle b_2 \rangle Y^{i_2} \dots \langle b_m \rangle Y^{i_m} & \\ \Rightarrow & b_1X[p_1] \dots [p_n][b_12]\$2Y^{i_0+1}ZY^{i_1}\langle b_2 \rangle Y^{i_2} \dots \langle b_m \rangle Y^{i_m} & [[b_12]] \\ \Rightarrow & b_1b_2X[p_1] \dots [p_n][b_12][b_22]\$2Y^{i_0+1}Y^{i_1+1}ZY^{i_2} \dots \langle b_m \rangle Y^{i_m} & [[b_22]] \\ \Rightarrow^{m-3} & b_1b_2 \dots b_{m-1}X[p_1] \dots [p_n][b_12] \dots [b_{m-1}2]\$2Y^{i_0+1}Y^{i_1+1} \dots & \\ & \dots Y^{i_{m-2}}ZY^{i_{m-1}}\langle b_m \rangle Y^{i_m} & [\bar{\sigma}] \\ \Rightarrow & b_1b_2 \dots b_m[p_1] \dots [p_n][b_12] \dots [b_{m-1}2][b_m3]\$3Y^{i_0+1}Y^{i_1+1} \dots Y^{i_m+1} & [[b_m3]] \end{array}$$

where $[p_1], \dots, [p_n] \in lab(G)$ are labels that denote productions introduced in 1-2, $\langle b_1 \rangle, \dots, \langle b_m \rangle \in \Phi$, $b_1, \dots, b_m \in \bar{T}$, $\bar{\sigma} = [b_32] \dots [b_{m-1}2]$, $i_0, i_1, \dots, i_m \geq 0$, $m = |s|$, where $s \in L(\bar{G})$ is a corresponding sentence of the SCG \bar{G} .

Proof. Notice that $occur(lhs([a2]), \{X\}) = occur(rhs([a2]), \{X\}) = 1$ and $occur(lhs([a2]), \{Y\}) = occur(rhs([a2]), \{Y\}) = 1$. In every derivation step of $y \Rightarrow^* z$, the the first symbol $\langle b \rangle \in \Phi$, following Z is replaced with Z , X is changed to bX , and $\$2$ is changed to $l\$2$, where $l \in lab(G)$. As $[a2]$ and $[a3]$ are the only production labels p satisfying $alph(\{lhs(p)\}) \cap \Phi \neq \emptyset$, $alph(\{rhs(p)\}) \cap \Phi = \emptyset$ and $rpos([a2], 3) = Z$, $rpos([a2], 4) = Z$, Z can replace only the first occurrence of

$\langle b \rangle \in \Phi$ behind Z to generate $w \in T^*$. Productions labeled with $[a_2]$ are used $m-1$ times. Thus, $y \Rightarrow^* z$ has the form

$$\begin{aligned}
 & X[p_1] \dots [p_n] \$2 Z Y^{i_0} \langle b_1 \rangle Y^{i_1} \langle b_2 \rangle Y^{i_2} \dots \langle b_m \rangle Y^{i_m} \\
 \Rightarrow & b_1 X[p_1] \dots [p_n] [b_1 2] \$2 Y^{i_0+1} Z Y^{i_1} \langle b_2 \rangle Y^{i_2} \dots \langle b_m \rangle Y^{i_m} \quad [[b_1 2]] \\
 \Rightarrow & b_1 b_2 X[p_1] \dots [p_n] [b_1 2] [b_2 2] \$2 Y^{i_0+1} Y^{i_1+1} Z Y^{i_2} \dots \langle b_m \rangle Y^{i_m} \quad [[b_2 2]] \\
 \Rightarrow^{m-3} & b_1 b_2 \dots b_{m-1} X[p_1] \dots [p_n] [b_1 2] [b_{m-1} 2] \$2 Y^{i_0+1} Y^{i_1+1} \dots \\
 & \dots Y^{i_{m-2}} Z Y^{i_{m-1}} \langle b_m \rangle Y^{i_m} \quad [\bar{\sigma}]
 \end{aligned}$$

where every sentential form satisfies $\bar{T}^* \{X\} \text{lab}(G)^+ \{\$2\} \{Y\}^* \{Z\} (\Phi \cup \{Y\})^+$.

Finally, some production labeled with $[a_3]$ is applied; therefore, $z \Rightarrow u$ can be expressed as

$$\begin{aligned}
 & b_1 b_2 \dots b_{m-1} X[p_1] \dots [p_n] [b_1 2] \dots [b_{m-1} 2] \$2 Y^{i_0+1} Y^{i_1+1} \dots \\
 & \dots Y^{i_{m-2}} Z Y^{i_{m-1}} \langle b_m \rangle Y^{i_m} \\
 \Rightarrow & b_1 b_2 \dots b_m [p_1] \dots [p_n] [b_1 2] \dots [b_{m-1} 2] [b_m 3] \$3 Y^{i_0+1} Y^{i_1+1} \dots Y^{i_m+1} \quad [[b_m 3]]
 \end{aligned}$$

with $u \in \bar{T}^+ \text{lab}(G)^+ \{\$3\} \{Y\}^+$.

Putting together the previous parts of derivation, we obtain the formulation of Claim 3. Thus, Claim 3 holds. \square \square

Claim 4. In

$$\begin{aligned}
 u & \Rightarrow^+ v \quad [\tau] \\
 & \Rightarrow w \quad [[4]]
 \end{aligned}$$

of derivation (1), every sentential form s of $u \Rightarrow^+ v$ satisfies $s \in \bar{T}^+ \text{lab}(G)^+ \{\$3\} \{Y\}^*$ and $w \in \bar{T}^+ \text{lab}(G)^+$. In greater detail, this derivation can be expressed as

$$\begin{aligned}
 & b_1 \dots b_m [p_1] \dots [p_n] \{\$3\} Y^i \\
 \Rightarrow & b_1 \dots b_m [p_1] \dots [p_n] [3] \{\$3\} Y^{i-1} \quad [[3]] \\
 \Rightarrow & b_1 \dots b_m [p_1] \dots [p_n] [3] [3] \{\$3\} Y^{i-2} \quad [[3]] \\
 \Rightarrow^{i-3} & b_1 \dots b_m [p_1] \dots [p_n] [3]^{i-1} \{\$3\} Y \quad [\bar{\tau}] \\
 \Rightarrow & b_1 \dots b_m [p_1] \dots [p_n] [3]^i \{\$3\} \quad [[3]] \\
 \Rightarrow & b_1 \dots b_m [p_1] \dots [p_n] [3]^i [4] \quad [[4]]
 \end{aligned}$$

where all $b_j \in \bar{T}$, $1 \leq j \leq m$ and $[p_k] \in \text{lab}(G)$, $1 \leq k \leq n$ are labels that denote productions introduced in steps 1 through 3 of the construction, $\bar{\tau}$ is a sequence of production labels $[3]$.

Proof. Notice that $\text{ipos}([3], 1) = \text{rpos}([3], 2) = \3 . Observe, that in order to generate $w \in T^*$ the first occurrence of Y following $\$3$ has to be taken by $[3]$ in each derivation step. Finally, $[4]$ is applied. At this moment, w satisfies $w \in T^*$ and $w \in \bar{T}^+ \text{lab}(G)^+$. \square

The next claim formally demonstrates how G generates the empty sentence ϵ followed by its parse.

Claim 5. G generates every $w \in L(G) \cap \text{lab}(G)^+$ in the following way

$$\begin{array}{rcl}
 S & \Rightarrow & [1_\epsilon] \$_1 \bar{S} \quad [[1_\epsilon]] \\
 & \Rightarrow^+ & x \quad [\rho] \\
 & \Rightarrow & y \quad [[2_\epsilon]] \\
 & \Rightarrow^+ & v \quad [\tau] \\
 & \Rightarrow & w \quad [[4]]
 \end{array} \tag{1}$$

where ρ and τ are sequences consisting from Ξ_1 and $\{\{3\}\}$, respectively.

Proof. Notice that $\text{alph}(\{w\}) \cap \bar{T} = \emptyset$ and only productions labeled with $p \in \Xi_3$ satisfy $X \in \text{alph}(\{\text{lhs}(p)\})$, $X \notin \text{alph}(\{\text{rhs}(p)\})$ and $X = {}_i\text{pos}(p, 1)$, $a = {}_\tau\text{pos}(p, 1)$, $a \in \bar{T}$. Therefore, X cannot appear in any sentential form of $S \Rightarrow^* w$, and the derivation starts with a step made by $[1_\epsilon]$. As $X \notin \text{alph}(\{x\})$ and for $p \in \Xi_2 \cup \Xi_3$, $X \in \text{alph}(\{\text{lhs}(p)\})$, the production labeled with $[2_\epsilon]$ has to be used. Observe that other derivation steps are made in the way described in Claim 2 and Claim 4. \square

From Claims 4 and 5, it follows that for every recursively enumerable language L , there exists a PSCG G such that G is a proper generator of its sentences with their parses and $L = \pi(L(G))$. \square

From Theorem 1, we obtain:

Corollary 1. *For every recursively enumerable language L , there exists a PSCG G such that G is a proper generator of its sentences with their parses and $L = L(G)/\text{lab}(G)^* \cap \text{alph}(L)^*$.*

Alternatively, we can introduce a SCG $G = (V, P, S, T)$, as a proper generator of its sentences preceded by their parses so that $L(G) = \{x \mid x = \rho y, y \in (T - \text{lab}(G))^*, \rho \in \text{lab}(G)^*, S \Rightarrow^* x[\rho]\}$.

Theorem 2. *For every recursively enumerable language L , there exists a PSCG G such that G is a proper generator of its sentences preceded by their parses and $L = \pi(L(G))$.*

Proof. This theorem can be proved by a straightforward modification of Theorem 1. A detailed version of this proof is left to the reader. \square

Corollary 2. *For every recursively enumerable language L , there exists a PSCG G such that G is a proper generator of its sentences preceded by their parses and $L = \text{lab}(G)^* \setminus L(G) \cap \text{alph}(L)^*$.*

5 Conclusion

In this concluding section, we make some final notes and suggestions regarding the future investigation.

First, notice that all the above results can be also established so that the generated sentences are followed by the reversals of their parses.

Second, consider the unordered scattered context grammars (see page 260 in [13]). In essence, in this version of scattered context grammars, we apply a production of the form $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n)$ so we simultaneously replace A_i with x_i , for all $i = 1, \dots, n$, no matter in what order the nonterminals A_i appear in the rewritten word. Naturally, we are tempted to use the construction given in the proof of Theorem 1 for these grammars in order to obtain analogical results to the above results. Unfortunately, this construction does not work for the unordered versions of scattered context grammars. Specifically, steps 3 and 4 of the construction require the prescribed order of rewritten nonterminals; otherwise, the result is not guaranteed. Can we prove the results of this paper in terms of unordered scattered context grammars by using some other methods?

Finally, let us recall that we have demonstrated that for every recursively enumerable language, there exists a propagating scattered context grammar that generate the language's sentences followed by their parses. From a broader perspective, we could naturally reformulate this generation of sentences with their parses in terms of other propagating rewriting mechanisms that define the language family contained in the family of context-sensitive languages. Probably, some propagating parallel rewriting mechanisms, such as propagating PC grammar systems (see Chapter 4 in Volume 2 of [12]), can be used in this way. Furthermore, some propagating regulated grammars, such as propagating matrix grammars (see Chapter 3 in Volume 3 of [12]), seems to be suitable for this generation as well. On the other hand, we can hardly base the generation of sentences with their parses upon classical sequential rewriting mechanisms, such as context-free grammars. The authors suggest these problem areas as the topics of future investigation that continues with the discussion opened in the present paper.

Acknowledgements We thank the anonymous referee for useful comments concerning the first version of this paper. The first author gladly acknowledges support of GACR grant 201/04/0441.

References

- [1] Chatterjee, S. (eds.): *Languages and Compilers for Parallel Computing*, Springer-Verlag, London, 1999.
- [2] Darte, A. et al. (eds.): *Compilers for Parallel Computers*, World Scientific, Singapore, 2000.
- [3] Fernau, H.: Scattered Context Grammars with Regulation, *Annals of Bucharest Univ., Math.-Informatics Series* 45(1) (1996), 41–49.
- [4] Gonczarowski, J. and Warmuth, M. K.: Scattered Versus Context-Sensitive Rewriting, *Acta Informatica* 27 (1989), 81–95.
- [5] Greibach, S. and Hopcroft, J. E.: Scattered Context Grammars, *Journal of Computer and System Sciences* 3 (1969), 233–247.

- [6] Meduna, A.: *Automata and Languages: Theory and Applications*, Springer-Verlag, London, 2000.
- [7] Meduna, A.: A Trivial Method of Characterizing the Family of Recursively Enumerable Languages by Scattered Context Grammars, *EATCS Bulletin* 56 (1995), 104–106.
- [8] Meduna, A.: Generative Power of Three-Nonterminal Scattered Context Grammars, *Theoretical Computer Science* 237 (2000), 625–631.
- [9] Midkiff, S. P. et al. (eds.): *Languages and Compilers for Parallel Computing* (13th International Workshop on Languages and Compilers for Parallel Computing, 2000, Yorktown Heights, N.Y.), Springer, London, 2001.
- [10] Rauchwerger, L. (eds.): *Languages and Compilers for Parallel Computing* (16th International Workshop, October 2003, Colledge Station, Texas), Springer, London, 2004.
- [11] Revesz, G. E.: *Introduction to Formal Language Theory*, McGraw-Hill, New York, 1983.
- [12] Rozenberg, G. and Salomaa, A. (eds.): *Handbook of Formal Languages*, Volume 1 through 3, Springer-Verlag, 1997.
- [13] Salomaa, A.: *Formal Languages*, Academic Press, London, 1973.
- [14] Vaszil, G.: On the Number of Conditional Rules in Simple Semi-conditional Grammars, *Theoretical Computer Science*, 2004 (in press).
- [15] Virkkunen, V.: On Scattered Context Grammars, *Acta Universitatis Oulensis*, Series A, Mathematica 6 (1973), 75–82.
- [16] Wolfe, M. J.: *High Performance Compilers for Parallel Computing*, Addison-Wesley, Redwood City, 1996.

Received May, 2004

Varieties of Tree Languages Definable by Syntactic Monoids

Saeed Salehi *

Abstract

An algebraic characterization of the families of tree languages definable by syntactic monoids is presented. This settles a question raised by several authors.

1 Introduction

A Variety Theorem establishing a bijective correspondence between general varieties of tree languages definable by syntactic monoids and varieties of finite monoids, is proved. This has been a relatively long-standing open problem, the most recent references to which are made by Ésik [4] as “No variety theorem is known in the semigroup [monoid] approach” (page 759), and by Steinby [18] as “there are no general criteria for deciding whether or not a given GVTL [general variety of tree languages] can or cannot be defined by syntactic monoids” (page 41). The question was also mentioned in the last section of Wilke’s paper [21].

Most of the interesting classes of algebraic structures form varieties, and similarly, most of the interesting families of tree or string languages studied in the literature turn out to be varieties of some kind. The first Variety Theorem was proved by Eilenberg [3] who established a correspondence between varieties of finite monoids and varieties of regular (string) languages. It was motivated by characterizations of several families of languages by syntactic monoids or semigroups (see [3],[10]), above all by Schützenberger’s [15] theorem connecting star-free languages and aperiodic monoids.

Eilenberg’s theorem has since been extended in various directions. One could mention Pin’s [11] Variety Theorem for positive varieties of string languages and varieties of ordered monoids, or Thérien’s [19] extension that includes also varieties of congruences on free monoids. On the level of universal algebra, where tree automata and tree languages are studied, a Variety Theorem was proved by Steinby [16] for recognizable subsets of finitely generated free algebras. Both Eilenberg’s $*$ -varieties and $+$ -varieties, as well as varieties of regular tree languages (which was

*Turku Center for Computer Science, DataCity - Lemminkäisenkatu 14 A, FIN-20520 Turku, e-mail: saeed@cs.utu.fi

worked out in [17]), are special cases of the results of [16]. The correspondence to varieties of congruences, and some other generalizations, were added later by Almeida [1] and Steinby [17, 18]. Another example is Ésik's [4] Variety Theorem between tree languages and theories (see also [5]). As Ésik observes in [4], page 758: "The crucial concept in any 'Variety Theorem' is that of the 'syntactic structure' or 'syntactic algebra'." For almost all those syntactic structures associated to tree languages in the literature, one (or some) variety theorem(s) have been proved. The most famous 'syntactic structure' for which a variety theorem was not known, is the syntactic semigroup/monoid of a tree language, introduced by Thomas [20], and further studied by Salomaa [14]. A different formalism, based on the essentially same concept, was brought up by Nivat and Podelski [6], [13].

To establish our correspondence between varieties of tree languages and varieties of finite monoids, we add three more closure properties to the definition of a general tree language variety introduced in [18]. One of them, that of being closed under inverse tree homomorphisms, is already investigated by Ésik [4], and the other two are stated in Theorem 24.

2 Notation and Preliminaries

Our notation is mainly based on [18]. However for understanding our results it is not necessary to read the whole of [18]. Here, we list the terminology used throughout the paper.

A finite set of function symbols is called a *ranked alphabet*. If Σ is a ranked alphabet, for every $m \geq 0$, the set of m -ary function symbols of Σ is denoted by Σ_m . In particular, Σ_0 is the set of constant symbols of Σ . For a ranked alphabet Σ and a *leaf alphabet* X , the set of ΣX -trees $T(\Sigma, X)$ is the smallest set satisfying

- (1) $\Sigma_0 \cup X \subseteq T(\Sigma, X)$, and
- (2) $f(t_1, \dots, t_m) \in T(\Sigma, X)$, for all $f \in \Sigma_m$ ($m > 0$) and $t_1, \dots, t_m \in T(\Sigma, X)$.

Any subset of $T(\Sigma, X)$ is called a *tree language*.

The ΣX -term algebra $\mathcal{T}(\Sigma, X) = (T(\Sigma, X), \Sigma)$ is defined by setting

- (1) $c^{\mathcal{T}(\Sigma, X)} = c$ for each $c \in \Sigma_0$, and
- (2) $f^{\mathcal{T}(\Sigma, X)}(t_1, \dots, t_m) = f(t_1, \dots, t_m)$ for all $m > 0$, $f \in \Sigma_m$, and $t_1, \dots, t_m \in T(\Sigma, X)$.

Let ξ be a (special) symbol which does not appear in any ranked alphabet or leaf alphabet considered here. The set of ΣX -contexts, denoted by $C(\Sigma, X)$, consists of the $\Sigma(X \cup \{\xi\})$ -trees in which ξ appears exactly once. For $P, Q \in C(\Sigma, X)$ and $t \in T(\Sigma, X)$ the context $Q \cdot P$, the composite of P and Q , results from P by replacing the special leaf ξ with Q , and the term $t \cdot P$ results from P by replacing ξ with t . Note that $C(\Sigma, X)$ is a monoid with composition as the operation and ξ as the unit element, and that $t \cdot (Q \cdot P) = (t \cdot Q) \cdot P$ holds for all $P, Q \in C(\Sigma, X)$, $t \in T(\Sigma, X)$. For a tree language $T \subseteq T(\Sigma, X)$ and context P , the *inverse translation* of T under

P is $P^{-1}(T) = \{t \in T(\Sigma, X) \mid t \cdot P \in T\}$. Also the *inverse morphism* of T under a homomorphism $\varphi : T(\Sigma, Y) \rightarrow T(\Sigma, X)$ is $T\varphi^{-1} = \{t \in T(\Sigma, Y) \mid t\varphi \in T\}$.

A ΣX -recognizer (\mathcal{A}, α, F) consists of a finite Σ -algebra $\mathcal{A} = (A, \Sigma)$, an *initial assignment* $\alpha : X \rightarrow A$, and a *set of final states* $F \subseteq A$. The function α can uniquely be extended to a homomorphism $\alpha^{\mathcal{A}} : T(\Sigma, X) \rightarrow \mathcal{A}$, and the tree language recognized by (\mathcal{A}, α, F) is $\{t \in T(\Sigma, X) \mid t\alpha^{\mathcal{A}} \in F\}$. In that case we also simply say that T is recognized by the algebra \mathcal{A} .

All algebras considered in this paper, except for term algebras, are finite, and the tree languages studied here are recognizable by finite algebras. A class of finite algebras of a fixed type is called a *variety of finite algebras* if it is closed under subalgebras, homomorphic images, and finite products. They are sometimes called *pseudo-varieties*, to be differentiated from real varieties whose members need not to be finite. Birkhoff's variety theorem [2] provides a logical characterization of those "original" varieties. In particular, a variety of finite monoids, abbreviated by VFM, is a class of finite monoids closed under submonoids, homomorphic images, and finite monoid products. A family $\mathcal{V} = \{\mathcal{V}(X)\}$ of tree languages of a fixed type Σ is a mapping which assigns to every finite leaf alphabet a collection $\mathcal{V} = \{\mathcal{V}(X)\}$ of recognizable ΣX -tree languages. A family \mathcal{V} is called a *variety of tree languages* if each $\mathcal{V}(X)$ is closed under Boolean operations and inverse translations, and the whole collection is closed under the inverse homomorphisms between term algebras (see [17]; below we will consider generalized varieties of tree languages).

Let $\mathcal{A} = (A, \Sigma)$ be an algebra. Every elementary context

$$P = f(a_1, \dots, \xi, \dots, a_m) \in C(\Sigma, A),$$

where $f \in \Sigma_m$ and $a_1, \dots, a_m \in A$, induces a unary function on A defined by $P^{\mathcal{A}}(a) = f^{\mathcal{A}}(a_1, \dots, a, \dots, a_m)$ for each $a \in A$. Such functions are called *elementary translations* of \mathcal{A} . The functions induced by compositions of such elementary contexts are defined by setting $(Q \cdot P)^{\mathcal{A}}(a) = P^{\mathcal{A}}(Q^{\mathcal{A}}(a))$ for any two contexts P and Q and any $a \in A$. These functions constitute the set of *translations* of \mathcal{A} denoted by $\text{Tr}(\mathcal{A})$. Note that two different contexts may induce the same translation.

The set $\text{Tr}(\mathcal{A})$ is a monoid with composition as the operation, called the *translation monoid* of \mathcal{A} , which is also denoted by $\text{Tr}(\mathcal{A})$. We note that $\text{Tr}(\mathcal{A})$ includes the identity translation $\xi^{\mathcal{A}} = 1_A$. The composition of translations p and q is denoted by $q \cdot p$, that is $(q \cdot p)(a) = p(q(a))$ for all $a \in A$ (cf. Section 5 of [18]).

For a tree language $T \subseteq T(\Sigma, X)$, the *syntactic congruence* θ_T of T is defined by

$$t \theta_T s \iff \forall P \in C(\Sigma, X) (t \cdot P \in T \leftrightarrow s \cdot P \in T),$$

for $t, s \in T(\Sigma, X)$, and the *syntactic algebra* $\text{SA}(T)$ of T is the quotient Σ -algebra $T(\Sigma, X)/\theta_T$ (see Definition 5.9 of [18]).

Also, the *m-congruence* μ_T of T on the monoid $C(\Sigma, X)$ is defined by

$$P \mu_T Q \iff \forall R \in C(\Sigma, X) \forall t \in T(\Sigma, X) (t \cdot P \cdot R \in T \leftrightarrow t \cdot Q \cdot R \in T),$$

for $P, Q \in C(\Sigma, X)$, and the *syntactic monoid* $\text{SM}(T)$ of T is the quotient monoid $C(\Sigma, X)/\mu_T$ (cf. [20] or Definition 10.1 of [18]).

Remark 1. It was shown in [14] that the translation monoid of the syntactic algebra of a tree language is isomorphic to the syntactic monoid of the tree language, i.e., $\text{Tr}(\text{SA}(T)) \cong \text{SM}(T)$ for every tree language T .

A *tree homomorphism* is a mapping $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ for ranked alphabets Σ and Ω , and leaf alphabets X and Y , determined by some mappings $\varphi_X : X \rightarrow T(\Omega, Y)$, and $\varphi_m : \Sigma_m \rightarrow T(\Omega, Y \cup \{\xi_1, \dots, \xi_m\})$, where $\Sigma_m \neq \emptyset$ and the ξ_i 's are new variables, inductively as follows

- (1) $x\varphi = \varphi_X(x)$ for $x \in X$, $c\varphi = \varphi_0(c)$ for $c \in \Sigma_0$, and
- (2) $f(t_1, \dots, t_n)\varphi = \varphi_n(f)[\xi_1 \leftarrow t_1\varphi, \dots, \xi_n \leftarrow t_n\varphi]$ that is ξ_i is replaced with $t_i\varphi$ for all i (cf. [18], page 7).

A tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is called *regular* if for every $f \in \Sigma_m$ ($m \geq 1$), each ξ_1, \dots, ξ_m appears exactly once in $\varphi_m(f)$.

The unique extension $\varphi_* : C(\Sigma, X) \rightarrow C(\Omega, Y)$ of a regular tree homomorphism φ to contexts is obtained by setting $\varphi_*(\xi) = \xi$ (cf. [18], Proposition 10.3).¹ We note that the identities $(Q \cdot P)\varphi_* = Q\varphi_* \cdot P\varphi_*$ and $(t \cdot Q \cdot P)\varphi = t\varphi \cdot Q\varphi_* \cdot P\varphi_*$ hold for all $P, Q \in C(\Sigma, X)$ and $t \in T(\Sigma, X)$.

3 Algebras Definable by Translation Monoids

The notions of *subalgebra*, *homomorphism*, and *direct product* are defined as usual in Universal Algebra, whereas for their generalizations, *g-subalgebra*, *g-homomorphism*, and *generalized product*, are defined for algebras which are not necessarily of the same type. We recall the following definitions from [18] (Definitions 3.1, 3.2, 3.3, 3.14).

Definition 2. Let $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Omega)$ be finite algebras.

The algebra \mathcal{B} is a *g-subalgebra* of \mathcal{A} , in notation $\mathcal{B} \subseteq_g \mathcal{A}$, if $B \subseteq A$, $\Omega_m \subseteq \Sigma_m$ for all $m \geq 0$, and for every $g \in \Omega_m$, $g^{\mathcal{B}}$ is the restriction of $g^{\mathcal{A}}$ to B .

An *assignment* is a mapping $\kappa : \Sigma \rightarrow \Omega$ such that $\kappa(\Sigma_m) \subseteq \Omega_m$ for all $m \geq 0$.

A *g-morphism* from \mathcal{A} to \mathcal{B} is a pair (κ, φ) , where $\kappa : \Sigma \rightarrow \Omega$ is an assignment and $\varphi : A \rightarrow B$ is a mapping satisfying $f^{\mathcal{A}}(a_1, \dots, a_m)\varphi = (f\kappa)^{\mathcal{B}}(a_1\varphi, \dots, a_m\varphi)$ for any $m \geq 0$; $f \in \Sigma_m$, and $a_1, \dots, a_m \in A$. If both κ and φ are surjective, then (κ, φ) is called a *g-epimorphism*, and in that case we write $\mathcal{B} \leftarrow_g \mathcal{A}$ (\mathcal{B} is a g-epimorphic image of \mathcal{A}). When \mathcal{B} is a g-epimorphic image of a g-subalgebra of \mathcal{A} , we write $\mathcal{B} \prec_g \mathcal{A}$. When both κ and φ are bijective, (κ, φ) is called a *g-isomorphism*, and $\mathcal{B} \cong_g \mathcal{A}$ means that \mathcal{B} and \mathcal{A} are g-isomorphic.

Let $\Sigma^1, \dots, \Sigma^n$ and Γ be ranked alphabets. The product $\Sigma^1 \times \dots \times \Sigma^n$ is a ranked alphabet such that $(\Sigma^1 \times \dots \times \Sigma^n)_m = \Sigma_m^1 \times \dots \times \Sigma_m^n$ for every $m \geq 0$. For any assignment $\kappa : \Gamma \rightarrow \Sigma^1 \times \dots \times \Sigma^n$, and any algebras $\mathcal{A}_1 = (A_1, \Sigma^1), \dots, \mathcal{A}_n = (A_n, \Sigma^n)$, the κ -*product* of $\mathcal{A}_1, \dots, \mathcal{A}_n$ is the Γ -algebra $\kappa(\mathcal{A}_1, \dots, \mathcal{A}_n) = (A_1 \times \dots \times A_n, \Gamma)$ defined by

- (1) $c^{\kappa(\mathcal{A}_1, \dots, \mathcal{A}_n)} = (c_1^{\mathcal{A}_1}, \dots, c_n^{\mathcal{A}_n})$ for $c \in \Gamma_0$, where $c\kappa = (c_1, \dots, c_n)$, and

¹Indeed any tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ can be extended to $\bar{\varphi} : C(\Sigma, X) \rightarrow T(\Omega, Y \cup \{\xi\})$ by setting $\xi\bar{\varphi} = \xi$, but if φ is not regular the range of $\bar{\varphi}$ may not be $C(\Omega, Y)$. Hence the regularity of φ is needed for the existence of the extension φ_* , see also Example 18.

$$(2) f^{\kappa(\mathcal{A}_1, \dots, \mathcal{A}_n)}(\mathbf{a}_1, \dots, \mathbf{a}_m) = (f_1^{\mathcal{A}_1}(a_{11}, \dots, a_{m1}), \dots, f_n^{\mathcal{A}_n}(a_{1n}, \dots, a_{mn}))$$

for $f \in \Gamma_m$ ($m > 0$) and $\mathbf{a}_i = (a_{i1}, \dots, a_{in}) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_n$, where $f\kappa = (f_1, \dots, f_n)$.

Without specifying the assignment κ , such algebras are called *g-products*.

In the notations \subseteq_g , \leftarrow_g , \prec_g , and \cong_g , the subscript g is dropped when \mathcal{A} and \mathcal{B} are of the same type, say Σ , and the assignment $\kappa : \Sigma \rightarrow \Sigma$ is the identity mapping.

The abbreviation GVFA stands for *general variety of finite algebras* which is a class of finite algebras, of all finite types, closed under g-sub-algebras, g-epimorphic images, and g-products (Definition 4.3 of [18]). It is easy to see that a class of algebras \mathbf{K} is a GVFA, if for any $\mathcal{A}_1, \dots, \mathcal{A}_n \in \mathbf{K}$, any g-product $\kappa(\mathcal{A}_1, \dots, \mathcal{A}_n)$, and any algebra \mathcal{A} , if $\mathcal{A} \prec_g \kappa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ then $\mathcal{A} \in \mathbf{K}$ (cf. Corollary 4.8 of [18]).

Definition 3. For a VFM \mathbf{M} , \mathbf{M}^a is the class of all finite algebras whose translation monoids are in \mathbf{M} , i.e., $\mathcal{A} \in \mathbf{M}^a \Leftrightarrow \text{Tr}(\mathcal{A}) \in \mathbf{M}$ for any finite algebra \mathcal{A} .

A class of finite algebras \mathbf{K} is said to be *definable by translation monoids*, if there is a VFM \mathbf{M} such that $\mathbf{M}^a = \mathbf{K}$.

By Proposition 10.8 of [18], a class of finite algebras definable by translation monoids is a GVFA. In fact, any such class can be proved to be a *d-variety of finite algebras* (see page 758 of [4]). An algebraic characterization of the classes of finite algebras definable by translation monoids is given in the main theorem of this section.

Definition 4. Let \mathcal{A} be a finite algebra. With each translation $p \in \text{Tr}(\mathcal{A})$ we associate a unary function symbol \bar{p} . Let $\Lambda_{\mathcal{A}} = \{\bar{p} \mid p \in \text{Tr}(\mathcal{A})\}$ be the unary ranked alphabet formed by these symbols and let the $\Lambda_{\mathcal{A}}$ -algebra $\mathcal{A}^e = (\text{Tr}(\mathcal{A}), \Lambda_{\mathcal{A}})$ be defined by $\bar{p}^{\mathcal{A}^e}(q) = q \cdot p$ for all $p, q \in \text{Tr}(\mathcal{A})$.

The proof of the main theorem of this section is based on the following lemmas (cf. [8, 9] for similar results for unary algebras).

Lemma 5. For any finite algebra \mathcal{A} , $\text{Tr}(\mathcal{A}) \cong \text{Tr}(\mathcal{A}^e)$.

Proof. The elementary translations of \mathcal{A}^e are of the form $\bar{p}^{\mathcal{A}^e}(\xi)$ where $p \in \text{Tr}(\mathcal{A})$, and clearly $\bar{q}^{\mathcal{A}^e}(\xi) \cdot \bar{p}^{\mathcal{A}^e}(\xi) = \bar{q \cdot p}^{\mathcal{A}^e}(\xi)$ for all $q, p \in \text{Tr}(\mathcal{A})$. For the identity translation $1_{\mathcal{A}}$ of \mathcal{A} the translation $\bar{1}_{\mathcal{A}}^{\mathcal{A}^e}(\xi)$ is the identity translation of \mathcal{A}^e . This means that $\text{Tr}(\mathcal{A}^e) = \{\bar{p}^{\mathcal{A}^e}(\xi) \mid p \in \text{Tr}(\mathcal{A})\}$. Moreover, $\bar{p}^{\mathcal{A}^e}(\xi) \neq \bar{q}^{\mathcal{A}^e}(\xi)$ whenever $p \neq q$, since $\bar{p}^{\mathcal{A}^e}(\xi) = \bar{q}^{\mathcal{A}^e}(\xi)$ implies $p = 1_{\mathcal{A}} \cdot p = \bar{p}^{\mathcal{A}^e}(1_{\mathcal{A}}) = \bar{q}^{\mathcal{A}^e}(1_{\mathcal{A}}) = 1_{\mathcal{A}} \cdot q = q$. Hence, the mapping $\text{Tr}(\mathcal{A}) \rightarrow \text{Tr}(\mathcal{A}^e)$, $p \mapsto \bar{p}^{\mathcal{A}^e}(\xi)$ is a monoid isomorphism. \square

Lemma 6. Let $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Omega)$ be two finite algebras.

1. If $\text{Tr}(\mathcal{A}) \prec \text{Tr}(\mathcal{B})$, then $\mathcal{A}^e \prec_g \mathcal{B}^e$.
2. $\text{Tr}(\mathcal{A}) \times \text{Tr}(\mathcal{B}) \cong \text{Tr}(\kappa(\mathcal{A}^e, \mathcal{B}^e))$ for some g-product $\kappa(\mathcal{A}^e, \mathcal{B}^e)$.

Proof. 1. Suppose $\text{Tr}(\mathcal{A}) \leftarrow M \subseteq \text{Tr}(\mathcal{B})$ for some monoid M . Let $\Lambda_M = \{\bar{p} \in \Lambda_{\mathcal{B}} \mid p \in M\}$. Then clearly $\mathcal{M} = (M, \Lambda_M) \subseteq_g \mathcal{B}^e$, where \mathcal{M} is defined by $\bar{p}^{\mathcal{M}}(q) = q \cdot p$ ($p, q \in M$). Let $\varphi : M \rightarrow \text{Tr}(\mathcal{A})$ be a monoid epimorphism. Define the assignment $\kappa : \Lambda_M \rightarrow \Lambda_{\mathcal{A}}$ by $\bar{q}\kappa = \overline{q\varphi}$ for all $q \in M$. It is clear that κ is surjective and for all $q, r \in M \subseteq \text{Tr}(\mathcal{B})$, $(\bar{q}^{\mathcal{B}^e}(r))\varphi = (r \cdot q)\varphi = r\varphi \cdot q\varphi = \overline{q\varphi}^{\mathcal{A}^e}(r\varphi) = (q\kappa)^{\mathcal{A}^e}(r\varphi)$. Hence $(\kappa, \varphi) : \mathcal{M} \rightarrow \mathcal{A}^e$ is a g-epimorphism. Thus $\mathcal{A}^e \leftarrow_g \mathcal{M} \subseteq_g \mathcal{B}^e$.

2. Let $\Gamma = \{\langle p, q \rangle \mid p \in \text{Tr}(\mathcal{A}), q \in \text{Tr}(\mathcal{B})\}$ be a set of unary function symbols, and define the assignment $\kappa : \Gamma \rightarrow \Lambda_{\mathcal{A}} \times \Lambda_{\mathcal{B}}$ by $\langle p, q \rangle \kappa = (\bar{p}, \bar{q})$. Let $\mathcal{P} = \kappa(\mathcal{A}^e, \mathcal{B}^e)$ be the corresponding g-product of \mathcal{A}^e and \mathcal{B}^e . We show that $\text{Tr}(\mathcal{P}) = \{\langle p, q \rangle^{\mathcal{P}}(\xi) \mid p \in \text{Tr}(\mathcal{A}), q \in \text{Tr}(\mathcal{B})\}$. Firstly, we note that if $1_{\mathcal{A}}$ and $1_{\mathcal{B}}$ are the identity translations of \mathcal{A} and \mathcal{B} respectively, then $\langle 1_{\mathcal{A}}, 1_{\mathcal{B}} \rangle^{\mathcal{P}}(\xi)$ is the identity translation of \mathcal{P} . Secondly, by the definition of κ -products, for all $p, p' \in \text{Tr}(\mathcal{A})$, $q, q' \in \text{Tr}(\mathcal{B})$,

$$\langle p, q \rangle^{\mathcal{P}}(p', q') = (\bar{p}^{\mathcal{A}^e}(p'), \bar{q}^{\mathcal{B}^e}(q')) = (p' \cdot p, q' \cdot q).$$

Hence, if $\langle p, q \rangle^{\mathcal{P}}(\xi) = \langle p', q' \rangle^{\mathcal{P}}(\xi)$, then $(p, q) = (1_{\mathcal{A}} \cdot p, 1_{\mathcal{B}} \cdot q) = \langle p, q \rangle^{\mathcal{P}}(1_{\mathcal{A}}, 1_{\mathcal{B}}) = \langle p', q' \rangle^{\mathcal{P}}(1_{\mathcal{A}}, 1_{\mathcal{B}}) = (1_{\mathcal{A}} \cdot p', 1_{\mathcal{B}} \cdot q') = (p', q')$. So, $\langle p, q \rangle^{\mathcal{P}}(\xi) \neq \langle p', q' \rangle^{\mathcal{P}}(\xi)$, when $p \neq p'$ or $q \neq q'$. Finally, we show that the set $\{\langle p, q \rangle^{\mathcal{P}}(\xi) \mid p \in \text{Tr}(\mathcal{A}), q \in \text{Tr}(\mathcal{B})\}$ is closed under the composition of translations.

For all $p, p', p'' \in \text{Tr}(\mathcal{A})$, $q, q', q'' \in \text{Tr}(\mathcal{B})$,

$$\begin{aligned} \langle p', q' \rangle^{\mathcal{P}} \cdot \langle p, q \rangle^{\mathcal{P}}(p'', q'') &= \langle p, q \rangle^{\mathcal{P}}(p'' \cdot p', q'' \cdot q') \\ &= \langle (p'' \cdot p') \cdot p, (q'' \cdot q') \cdot q \rangle \\ &= \langle p'' \cdot (p' \cdot p), q'' \cdot (q' \cdot q) \rangle \\ &= \langle p' \cdot p, q' \cdot q \rangle^{\mathcal{P}}(p'', q''). \end{aligned}$$

Hence, $\langle p', q' \rangle^{\mathcal{P}}(\xi) \cdot \langle p, q \rangle^{\mathcal{P}}(\xi) = \langle p' \cdot p, q' \cdot q \rangle^{\mathcal{P}}(\xi)$. It follows that the mapping $\text{Tr}(\mathcal{A}) \times \text{Tr}(\mathcal{B}) \rightarrow \text{Tr}(\mathcal{P})$, $(p, q) \mapsto \langle p, q \rangle^{\mathcal{P}}(\xi)$, is a monoid isomorphism. \square

Since g-products of g-products are g-isomorphic to a g-product of the original algebras (Lemma 4.2 of [18]), Lemma 6(2) can be generalized as follows.

Lemma 7. For any $n \geq 1$ and any algebras $\mathcal{A}_1, \dots, \mathcal{A}_n$ there is a g-product $\kappa(\mathcal{A}_1^e, \dots, \mathcal{A}_n^e)$ such that $\text{Tr}(\mathcal{A}_1) \times \dots \times \text{Tr}(\mathcal{A}_n) \cong \text{Tr}(\kappa(\mathcal{A}_1^e, \dots, \mathcal{A}_n^e))$.

Now we are ready to prove the main theorem.

Theorem 8. Any class of finite algebras \mathbf{K} is definable by translation monoids iff it is a GVFA such that $\mathcal{A} \in \mathbf{K}$ iff $\mathcal{A}^e \in \mathbf{K}$, for any \mathcal{A} .

Proof. Suppose $\mathbf{K} = \mathbf{M}^a$ for a VFM \mathbf{M} . Then by Lemma 5, $\text{Tr}(\mathcal{A}) \cong \text{Tr}(\mathcal{A}^e)$, so $\mathcal{A} \in \mathbf{K} \Leftrightarrow \text{Tr}(\mathcal{A}) \in \mathbf{M} \Leftrightarrow \text{Tr}(\mathcal{A}^e) \in \mathbf{M} \Leftrightarrow \mathcal{A}^e \in \mathbf{K}$. For the converse, suppose the GVFA \mathbf{K} satisfies the equivalence $\mathcal{A} \in \mathbf{K} \Leftrightarrow \mathcal{A}^e \in \mathbf{K}$ for any finite algebra \mathcal{A} . Let \mathbf{M} be the VFM generated by $\{\text{Tr}(\mathcal{A}) \mid \mathcal{A} \in \mathbf{K}\}$. We show that $\mathbf{K} = \mathbf{M}^a$. Obviously $\mathbf{K} \subseteq \mathbf{M}^a$. For the opposite inclusion, let $\mathcal{B} \in \mathbf{M}^a$. So, there are $\mathcal{A}_1, \dots, \mathcal{A}_m \in \mathbf{K}$

d-varieties of finite algebras and general tree language varieties closed under inverse tree homomorphisms. However, those varieties may not be definable by syntactic monoids, as the following example shows.

Example 13. Let $\text{Def}_1 = \{\text{Def}_1(\Sigma, X)\}$ be the family of 1-definite tree languages, i.e., $T \in \text{Def}_1(\Sigma, X)$ iff for all ΣX -trees t and s , $\text{root}(t) = \text{root}(s)$ and $t \in T$ imply $s \in T$, where $\text{root}(t)$ is the root symbol of t . It is a GVTL ([18]) which can be shown to be closed under inverse *strict* regular tree homomorphisms (see [4] Subsection 11.1 and Section 5 below). Let $\Sigma = \Sigma_2 = \{f, g\}$, $X = \{x, y\}$, and $T = \{x\} \cup \{f(t_1, t_2) \mid t_1, t_2 \in T(\Sigma, X)\}$. Clearly $T \in \text{Def}_1(\Sigma, X)$. It can be easily shown that the syntactic monoid of T consists of an identity element and two right zeros. This is also the syntactic monoid of the language T' of the ΣX -trees whose leftmost leaves are x , by Example 10.4 of [18]. Since $T' \notin \text{Def}_1(\Sigma, X)$, then Def_1 is not definable by syntactic monoids.

This actually shows that the GVTL of all definite tree languages is not definable by syntactic monoids, since T' is not k -definite for any $k \geq 1$.

Remark 14. In [7] it is claimed that the variety of definite tree languages can be characterized by the property that all the non-identity idempotents of their syntactic monoids are right zeros (left zeros in the formalism of [7]). This clearly stands in conflict with the above Example 13.

Indeed, it can be shown that Theorem 1 of [7] does not hold. When the syntactic semigroup of a tree language is defined as the syntactic monoid with the identity element removed, the authors clearly overlook the possibility that the identity element may be obtained also as the product of some non-identity elements, and the proof of the theorem of [7] holds in just one direction. A concrete example showing that the equality between lines 9 and 10 on page 189 does not necessarily hold, can be obtained by considering the tree language T' of our Example 13.

It can also be noted that finite monoids whose non-identity idempotents are right zeros, do not form a VFM. Finally, in Section 5 we shall see that a more appropriate definition of the syntactic semigroup and omitting trees that in a sense correspond to the empty word, does not save the result of [7].

We shall characterize the general varieties of tree languages that are definable by syntactic monoids by requiring them to satisfy two more conditions in addition to being closed under inverse regular tree homomorphisms.

Definition 15. A regular tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is said to be *full with respect to* a tree language $T \subseteq T(\Omega, Y)$, if for every $Q \in C(\Omega, Y)$ and every $s \in T(\Omega, Y)$, there are $P \in C(\Sigma, X)$ and $t \in T(\Sigma, X)$, such that $Q \mu_T P \varphi_*$ and $s \theta_T t \varphi$ hold.

Remark 16. At first glance it seems that verifying fullness of φ with respect to T requires checking the existence of $P \in C(\Sigma, X)$ and $t \in T(\Sigma, X)$ for all (infinitely many) $Q \in C(\Omega, Y)$ and $s \in T(\Omega, Y)$ such that $Q \mu_T P \varphi_*$ and $s \theta_T t \varphi$ hold. In fact it is decidable for a recognizable T to check whether or not φ is full with respect to T : let $\varphi^T : T(\Omega, Y) \rightarrow T(\Omega, Y)/\theta_T$, $t\varphi^T = t/\theta_T$ and

such that $\text{Tr}(\mathcal{B}) \prec \text{Tr}(\mathcal{A}_1) \times \cdots \times \text{Tr}(\mathcal{A}_m)$. By Lemma 7, $\text{Tr}(\mathcal{B}) \prec \text{Tr}(\mathcal{P})$ for some g-product \mathcal{P} of $\mathcal{A}_1^e, \dots, \mathcal{A}_m^e$. By the property of \mathbf{K} , $\mathcal{A}_1^e, \dots, \mathcal{A}_m^e \in \mathbf{K}$, and so $\mathcal{P} \in \mathbf{K}$, hence $\mathcal{P}^e \in \mathbf{K}$. By Lemma 6 (1) from $\text{Tr}(\mathcal{B}) \prec \text{Tr}(\mathcal{P})$ we get $\mathcal{B}^e \prec_g \mathcal{P}^e$, and since $\mathcal{P}^e \in \mathbf{K}$, also $\mathcal{B}^e \in \mathbf{K}$, which implies that $\mathcal{B} \in \mathbf{K}$. Thus $\mathbf{M}^a \subseteq \mathbf{K}$. \square

Remark 9. The proof of Theorem 8 also yields the fact that for any GVFA \mathbf{K} definable by translation monoids, the class $\{\text{Tr}(\mathcal{A}) \mid \mathcal{A} \in \mathbf{K}\}$ is a variety of finite monoids.

Another characterization of the classes of finite algebras definable by translation monoids which follows from Lemmas 5 and 6 is the following.

Theorem 10. Any class of finite algebras \mathbf{K} is definable by translation monoids iff it is a GVFA such that for all finite algebras \mathcal{A} and \mathcal{B} , if $\text{Tr}(\mathcal{A}) \cong \text{Tr}(\mathcal{B})$ and $\mathcal{A} \in \mathbf{K}$, then $\mathcal{B} \in \mathbf{K}$.

4 Families of Tree Languages Definable by Syntactic Monoids

A *general variety of tree languages* (GVTL) is a family $\mathcal{V} = \{\mathcal{V}(\Sigma, X)\}$ which assigns to every ranked alphabet Σ and leaf alphabet X , a set $\mathcal{V}(\Sigma, X)$ of recognizable ΣX -tree languages, and is closed under all Boolean operations, inverse translations, and inverse g-morphisms. That is to say, for any ranked alphabets Σ, Ω , leaf alphabets X, Y , context $P \in C(\Sigma, X)$, and g-morphism $\varphi : T(\Omega, Y) \rightarrow T(\Sigma, X)$ (see Definition 2), if $T, T' \in \mathcal{V}(\Sigma, X)$, then $T(\Sigma, X) \setminus T, T \cap T', P^{-1}(T) \in \mathcal{V}(\Sigma, X)$, and $T\varphi^{-1} \in \mathcal{V}(\Omega, Y)$ (Definition 7.1 of [18]).

For a family of recognizable tree languages \mathcal{V} , \mathcal{V}^a is the GVFA generated by the class $\{\text{SA}(T) \mid T \in \mathcal{V}(\Sigma, X), \text{ for some } \Sigma, X\}$.

Remark 11. The General Variety Theorem in [18], Proposition 9.15, implies that:

- (1) For any GVTL \mathcal{V} , the class \mathcal{V}^a satisfies the following equivalence for any tree language $T \subseteq T(\Sigma, X)$: $T \in \mathcal{V}(\Sigma, X) \Leftrightarrow \text{SA}(T) \in \mathcal{V}^a$.
- (2) For any GVFA \mathbf{K} there is a unique GVTL \mathcal{V} such that $\mathcal{V}^a = \mathbf{K}$.

Definition 12. For a VFM \mathbf{M} , let \mathbf{M}^t be the family of all recognizable tree languages whose syntactic monoids are in \mathbf{M} , that is to say for any tree language $T \subseteq T(\Sigma, X)$, $T \in \mathbf{M}^t(\Sigma, X) \Leftrightarrow \text{SM}(T) \in \mathbf{M}$ holds.

A family of recognizable tree languages \mathcal{V} is said to be *definable by syntactic monoids* if there is a VFM \mathbf{M} such that $\mathbf{M}^t = \mathcal{V}$.

Steinby has shown that for any VFM \mathbf{M} , \mathbf{M}^t is a GVTL ([18], Proposition 10.3). His proof can be applied to show that \mathbf{M}^t is also closed under inverse of regular tree homomorphisms. The general varieties of tree languages closed under inverse (arbitrary) tree homomorphisms are studied by Ésik [4] who characterized them by their *syntactic theories*. Theorem 14.2 of [4] establishes a correspondence between

$\lambda^T : C(\Omega, Y) \rightarrow C(\Omega, Y)/\mu_T$, $P\lambda^T = P/\mu_T$ be the natural morphisms. Then the tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is full with respect to T iff both the mappings $\varphi\varphi^T : T(\Sigma, X) \rightarrow T(\Omega, Y)/\theta_T$ and $\varphi_*\lambda^T : C(\Sigma, X) \rightarrow C(\Omega, Y)/\mu_T$ are surjective.

Recall that for an equivalence relation θ on a set A , the quotient set of A under θ is denoted by A/θ , and $a\theta$ is the equivalence θ -class containing $a \in A$.

Lemma 17. If $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is a regular tree homomorphism and $T \subseteq T(\Omega, Y)$, then $SM(T\varphi^{-1}) \prec SM(T)$, and if φ is full with respect to T , then $SM(T\varphi^{-1}) \cong SM(T)$.

Proof. We note that $\varphi_* : C(\Sigma, X) \rightarrow C(\Omega, Y)$ is a monoid homomorphism. Let $S \subseteq C(\Omega, Y)$ be the image of φ_* , and let μ be the restriction of μ_T to S . Then S/μ is a submonoid of $C(\Omega, Y)/\mu_T$. We show that $P\varphi_*\mu Q\varphi_*$ implies $P\mu_{T\varphi^{-1}}Q$ for all $P, Q \in C(\Sigma, X)$.

Suppose $P\varphi_*\mu Q\varphi_*$ and take arbitrary $t \in T(\Omega, Y)$ and $R \in C(\Omega, Y)$. Then

$$\begin{aligned} t \cdot P \cdot R \in T\varphi^{-1} &\Leftrightarrow t\varphi \cdot P\varphi_* \cdot R\varphi_* \in T \\ &\Leftrightarrow t\varphi \cdot Q\varphi_* \cdot R\varphi_* \in T \\ &\Leftrightarrow t \cdot Q \cdot R \in T\varphi^{-1}, \end{aligned}$$

that is $P\mu_{T\varphi^{-1}}Q$. So the mapping $\psi : S/\mu \rightarrow C(\Sigma, X)/\mu_{T\varphi^{-1}}$ defined by $((P\varphi_*)\mu)\psi = P\mu_{T\varphi^{-1}}$ is well-defined and surjective. It is also a monoid homomorphism, since $((P\varphi_*)\mu) \cdot ((Q\varphi_*)\mu)\psi = ((P \cdot Q)\varphi_*)\mu\psi = (P \cdot Q)\mu_{T\varphi^{-1}} = P\mu_{T\varphi^{-1}} \cdot Q\mu_{T\varphi^{-1}} = ((P\varphi_*)\mu)\psi \cdot ((Q\varphi_*)\mu)\psi$ for all $P, Q \in C(\Sigma, X)$. Hence $SM(T\varphi^{-1}) \leftarrow S/\mu \subseteq SM(T)$, so $SM(T\varphi^{-1}) \prec SM(T)$.

Now, suppose φ is full with respect to T . We show $P\mu_{T\varphi^{-1}}Q$ iff $P\varphi_*\mu_T Q\varphi_*$ for any $P, Q \in C(\Sigma, X)$. Clearly, $P\varphi_*\mu_T Q\varphi_*$ implies $P\mu_{T\varphi^{-1}}Q$. For the converse, suppose $P\mu_{T\varphi^{-1}}Q$, and take arbitrary $R' \in C(\Omega, Y)$, and $t' \in T(\Omega, Y)$. There are $R \in C(\Sigma, X)$ and $t \in T(\Sigma, X)$ such that $R\varphi_*\mu_T R'$ and $t\varphi\theta_T t'$. Hence

$$\begin{aligned} t' \cdot P\varphi_* \cdot R' \in T &\Leftrightarrow t\varphi \cdot P\varphi_* \cdot R\varphi_* \in T \\ &\Leftrightarrow (t \cdot P \cdot R)\varphi \in T \\ &\Leftrightarrow t \cdot P \cdot R \in T\varphi^{-1} \\ &\Leftrightarrow t \cdot Q \cdot R \in T\varphi^{-1} \\ &\Leftrightarrow t\varphi \cdot Q\varphi_* \cdot R\varphi_* \in T \\ &\Leftrightarrow t' \cdot Q\varphi_* \cdot R' \in T, \end{aligned}$$

which shows that $P\varphi_*\mu_T Q\varphi_*$. Hence $P\mu_{T\varphi^{-1}}Q$ iff $P\varphi_*\mu_T Q\varphi_*$, and since the function $\varphi_* : C(\Sigma, X) \rightarrow C(\Omega, Y)$ is a monoid homomorphism, the mapping $C(\Sigma, X)/\mu_{T\varphi^{-1}} \rightarrow C(\Omega, Y)/\mu_T$, $P\mu_{T\varphi^{-1}} \mapsto (P\varphi_*)\mu_T$ is a monoid isomorphism between $SM(T\varphi^{-1})$ and $SM(T)$. \square

In the following example we show that the regularity condition on φ in the previous lemma can not be relaxed.

Example 18. Define the ranked alphabets $\Omega = \Omega_2 = \{f\}$ and $\Sigma = \Sigma_1 = \{g, h\}$, and the leaf alphabet $X = \{u, v, w\}$. Let $(\mathbb{Z}_3, +)$ be the cyclic group of order 3. Define $\chi : T(\Omega, X) \rightarrow \mathbb{Z}_3$ inductively by $u\chi = 0$, $v\chi = 1$, $w\chi = 2$, and $f(t, s)\chi = t\chi + s\chi$. Let $T = \{0\}\chi^{-1}$. It is easy to see that the syntactic monoid of T consists of the μ_T -classes of the elementary contexts $f(u, \xi)$, $f(v, \xi)$, $f(w, \xi)$, and in fact $\text{SM}(T) \simeq (\mathbb{Z}_3, +)$.

Define the tree homomorphisms $\varphi, \psi : T(\Sigma, X) \rightarrow T(\Omega, X)$ by $\varphi_X(x) = \psi_X(x) = x$ for $x \in X$, and $\varphi_1(g) = \psi_1(g) = f(v, \xi)$, $\varphi_1(h) = f(\xi, \xi)$, and $\psi_1(h) = u$. These tree homomorphisms are not regular: ξ appears twice in $\varphi_1(h)$ and does not appear at all in $\psi_1(h)$.

We show that neither $\text{SM}(T\varphi^{-1})$ nor $\text{SM}(T\psi^{-1})$ can divide $\text{SM}(T)$. The following identities can be verified by straightforward computations:

$$\begin{aligned} &-(v \cdot h(\xi) \cdot g(\xi))\varphi\chi = 0, \quad (v \cdot g(\xi) \cdot h(\xi))\varphi\chi = 1, \text{ and} \\ &-(v \cdot h(\xi) \cdot g(\xi))\psi\chi = 1, \quad (v \cdot g(\xi) \cdot h(\xi))\psi\chi = 0. \end{aligned}$$

So, $(h(\xi) \cdot g(\xi), g(\xi) \cdot h(\xi)) \notin \mu_{T\varphi^{-1}}, \mu_{T\psi^{-1}}$ which proves that $\text{SM}(T\varphi^{-1})$ and $\text{SM}(T\psi^{-1})$ are not commutative.

Remark 19. Let \mathbf{C} be the variety of finite commutative monoids. By Example 18, the GVTL \mathbf{C}^t is not closed under inverse non-regular tree homomorphisms; cf. Theorem 24. So, \mathbf{C}^t is not definable by syntactic theories in the sense of [4]. On the other hand, by Example 13, the family of definite tree languages is not definable by syntactic monoids, even though it is definable by syntactic theories, cf. [4] Subsection 11.1.

Thus, the concepts of “definability by syntactic theories” and of “definability by syntactic monoids” are not comparable to each other, though they are both weaker than “definability by syntactic algebras”.

Lemma 20. Let $\mathcal{A} = (A, \Sigma)$ be a finite algebra, and X be a leaf alphabet disjoint from A . For any tree language $L \subseteq T(\Lambda_{\mathcal{A}}, X)$ recognized by \mathcal{A}^e , there exists a regular tree homomorphism $\varphi : T(\Lambda_{\mathcal{A}}, X) \rightarrow T(\Sigma, X \cup A)$, and a tree language $T \subseteq T(\Sigma, X \cup A)$ such that $L = T\varphi^{-1}$, and T can be recognized by a finite power \mathcal{A}^n where $n = |A|$.

Proof. Let $\alpha : X \rightarrow \text{Tr}(\mathcal{A})$ be an initial assignment for \mathcal{A}^e and $F \subseteq \text{Tr}(\mathcal{A})$ be a subset such that $L = \{t \in T(\Lambda_{\mathcal{A}}, X) \mid t\alpha^{\mathcal{A}^e} \in F\}$. Define the tree homomorphism $\varphi : T(\Lambda_{\mathcal{A}}, X) \rightarrow T(\Sigma, X \cup A)$ by $\varphi_X(x) = x$ for all $x \in X$, and for every $p \in \text{Tr}(\mathcal{A})$ choose a $\varphi_1(\bar{p}) \in C(\Sigma, A)$ such that $\varphi_1(\bar{p})^{\mathcal{A}} = p$. Obviously φ is a regular tree homomorphism. Suppose that $A = \{a_1, \dots, a_n\}$. Let $F' = \{(p(a_1), \dots, p(a_n)) \in A^n \mid p \in F\}$, and define the initial assignment $\beta : X \cup A \rightarrow A^n$ for \mathcal{A}^n by $x\beta = ((x\alpha)(a_1), \dots, (x\alpha)(a_n))$ for all $x \in X$, and $a\beta = (a, \dots, a) \in A^n$ for all $a \in A$. Let T be the subset of $T(\Sigma, X \cup A)$ recognized by $(\mathcal{A}^n, \beta, F')$. We show that $L = T\varphi^{-1}$. Every tree w in $T(\Lambda_{\mathcal{A}}, X)$ is of the form $w = \overline{p_1}(\overline{p_2}(\dots \overline{p_k}(x) \dots))$ for some $p_1, \dots, p_k \in \text{Tr}(\mathcal{A})$ ($k \geq 0$) and $x \in X$. For such a tree w ,

$$w\alpha^{\mathcal{A}^e} = x\alpha \cdot p_k \cdot \dots \cdot p_2 \cdot p_1, \text{ and}$$

$(w\varphi)\beta^{\mathcal{A}^n} = (x\alpha \cdot p_k \cdot \dots \cdot p_2 \cdot p_1(a_1), \dots, x\alpha \cdot p_k \cdot \dots \cdot p_2 \cdot p_1(a_n))$. So,

$$\begin{aligned}
 w\varphi \in T & \Leftrightarrow (w\varphi)\beta^{\mathcal{A}^n} \in F' \\
 & \Leftrightarrow \text{for some } p \in F, p(a) = x\alpha \cdot p_k \cdot \dots \cdot p_2 \cdot p_1(a) \text{ for all } a \in A \\
 & \Leftrightarrow x\alpha \cdot p_k \cdot \dots \cdot p_2 \cdot p_1 \in F \\
 & \Leftrightarrow w\alpha^{\mathcal{A}^e} \in F \\
 & \Leftrightarrow w \in L.
 \end{aligned}$$

□

Lemma 21. Let $\mathcal{A} = (A, \Sigma)$ be a finite algebra and X be a leaf alphabet disjoint from $A \cup \Sigma$. For any tree language $T \subseteq T(\Sigma, X)$ recognized by \mathcal{A} there exists a unary ranked alphabet Λ , and a regular tree homomorphism $\varphi : T(\Lambda, X \cup \Sigma_0) \rightarrow T(\Sigma, X)$ such that φ is full with respect to T , and for every $z \in X \cup \Sigma_0$, $T\varphi^{-1} \cap T(\Lambda, \{z\})$ can be recognized as a subset of $T(\Lambda, \{z\})$ by \mathcal{A}^e .

Proof. Let $\mathcal{B} = (B, \Sigma)$ be the syntactic algebra of T . Then $\mathcal{B} \prec \mathcal{A}$. Suppose $T = \{t \in T(\Sigma, X) \mid t\beta^B \in F\}$, where $\beta : X \rightarrow B$ is an initial assignment for \mathcal{B} and $F \subseteq B$. Since \mathcal{B} is the minimal tree automaton recognizing T , the set B is generated by $\beta(X)$. The mapping $\beta : X \rightarrow B$ can be uniquely extended to a monoid homomorphism $\beta_c : C(\Sigma, X) \rightarrow C(\Sigma, B)$. Since B is generated by $\beta(X)$, the mapping $\beta_c^B : C(\Sigma, X) \rightarrow \text{Tr}(\mathcal{B})$, $\beta_c^B(Q) = \beta_c(Q)^B$ is surjective. Define the tree homomorphism $\varphi : T(\Lambda_B, X \cup \Sigma_0) \rightarrow T(\Sigma, X)$ by $\varphi_X(x) = x$ for all $x \in X \cup \Sigma_0$, and for every $q \in \text{Tr}(\mathcal{B})$ choose a $\varphi_1(\bar{q}) = Q \in C(\Sigma, X)$ such that $\beta_c(Q)^B = q$. Note that φ is a regular tree homomorphism. It remains to show that φ is full with respect to T and that for every $z \in X \cup \Sigma_0$, $L_z = T\varphi^{-1} \cap T(\Lambda, \{z\})$ can be recognized as a subset of $T(\Lambda, \{z\})$ by \mathcal{B}^e . This will finish the proof since $\text{Tr}(\mathcal{B}) \prec \text{Tr}(\mathcal{A})$ follows from $\mathcal{B} \prec \mathcal{A}$ by Lemma 10.7 of [18], and so $\mathcal{B}^e \prec \mathcal{A}^e$ by Lemma 6, which implies that L_z can also be recognized by \mathcal{A}^e .

Firstly, we show that φ is full with respect to T . Let $Q \in C(\Sigma, X)$ be a context. For $q = \beta_c(Q)^B \in \text{Tr}(\mathcal{B})$, $\bar{q}(\xi)\varphi_* \mu_T Q$ holds. By induction on the height of t we show that for any $t \in T(\Sigma, X)$ there is an $s \in T(\Lambda_B, X \cup \Sigma_0)$ such that $t \theta_T s \varphi$. If $t = x \in X \cup \Sigma_0$, then $s \varphi \theta_T t$ for $s = t$. If $t = t' \cdot P$ for some $P \in C(\Sigma, X)$ and $t' \in T(\Sigma, X)$ such that the height of t' is less than the height of t , then by the induction hypothesis there is an $s' \in T(\Lambda_B, X \cup \Sigma_0)$ such that $t' \theta_T s' \varphi$. Also, for some $p \in \text{Tr}(\mathcal{B})$, $\bar{p}(\xi)\varphi_* \mu_T P$ holds. Let $s = \bar{p}(s')$. Then

$$s\varphi = s'\varphi \cdot \bar{p}(\xi)\varphi_* \theta_T t' \cdot P = t.$$

Secondly, we show that L_z can be recognized by \mathcal{B}^e for a fixed $z \in X \cup \Sigma_0$. Let 1_B be the identity translation of \mathcal{B} . Define the initial assignment $\alpha : \{z\} \rightarrow \text{Tr}(\mathcal{B})$ for \mathcal{B}^e by $z\alpha = 1_B$, and let $F_z = \{q \in \text{Tr}(\mathcal{B}) \mid q(z\beta^B) \in F\}$. We show that L_z is recognized by $(\mathcal{B}^e, \alpha, F_z)$. Every $w \in T(\Lambda_B, \{z\})$ can be written in the form

$$w = \bar{q}_1(\bar{q}_2(\dots \bar{q}_h(z)\dots))$$

for some $q_1, \dots, q_h \in \text{Tr}(\mathcal{B})$ ($h \geq 0$). For such a tree w ,

$w\alpha^{B^o} = 1_B \cdot q_h \cdot \dots \cdot q_2 \cdot q_1$, and $(w\varphi)\beta^B = q_h \cdot \dots \cdot q_2 \cdot q_1(z\beta^B)$. Thus,

$$\begin{aligned} w \in L_z &\Leftrightarrow w\varphi \in T &\Leftrightarrow (w\varphi)\beta^B \in F \\ &&\Leftrightarrow q_h \cdot \dots \cdot q_2 \cdot q_1(z\beta^B) \in F \\ &&\Leftrightarrow q_h \cdot \dots \cdot q_2 \cdot q_1 \in F_z \\ &&\Leftrightarrow w\alpha^{B^o} \in F_z. \end{aligned}$$

So, $L_z = \{w \in T(\Lambda, \{z\}) \mid w\alpha^{B^o} \in F_z\}$. □

We end the section by proving a Variety Theorem for tree languages and syntactic monoids, and presenting some examples that justify the theorem (another interesting example is presented in [12]).

Before presenting the main theorem we note two remarks.

Remark 22. Let Λ be a unary ranked alphabet. For every leaf alphabet X and every subset $Y \subseteq X$, $C(\Lambda, Y) = C(\Lambda, X)$, and the relation μ_T for a tree language $T \subseteq T(\Lambda, Y)$ on $C(\Lambda, Y)$ is the same relation μ_T on $C(\Lambda, X)$ when T is viewed as a subset of $T(\Lambda, X)$.

So, if a family of tree languages $\mathcal{V} = \{\mathcal{V}(\Sigma, X)\}$ is definable by syntactic monoids, then for every unary ranked alphabet Λ , and any leaf alphabets X and Y , if $Y \subseteq X$ then $\mathcal{V}(\Lambda, Y) \subseteq \mathcal{V}(\Lambda, X)$.

Recall the notion of \mathcal{V}^a at the beginning of the section.

Remark 23. By Propositions 6.13 and 5.8(b) of [18] it follows that every finite algebra can be represented as a subdirect product of the syntactic algebras of some tree languages that are recognizable by the algebra. This implies that for any GVTL \mathcal{V} and any finite algebra \mathcal{A} , if every tree language recognizable by \mathcal{A} belongs to \mathcal{V} , then $\mathcal{A} \in \mathcal{V}^a$.

Theorem 24. A family of recognizable tree languages \mathcal{V} is definable by syntactic monoids iff \mathcal{V} is a GVTL that is closed under inverse regular tree homomorphisms and satisfies the following conditions:

- (1) For every unary ranked alphabet Λ , and any leaf alphabets X and Y , if $Y \subseteq X$ then $\mathcal{V}(\Lambda, Y) \subseteq \mathcal{V}(\Lambda, X)$.
- (2) For any regular tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ which is full with respect to a tree language $T \subseteq T(\Omega, Y)$, if $T\varphi^{-1} \in \mathcal{V}(\Sigma, X)$ then $T \in \mathcal{V}(\Omega, Y)$.

Proof. That for any VFM M , M^* satisfies the conditions of Theorem 24 follows from Lemma 17, Remark 22, and the facts mentioned at the beginning of the section. For the converse, suppose the GVTL \mathcal{V} satisfies the conditions presented in the theorem. We complete the proof of the theorem by showing that \mathcal{V}^a satisfies the condition of Theorem 8. Indeed, Theorem 8 implies then that there is a VFM M such that $\mathcal{V}^a = M^a$, and

$$T \in \mathcal{V} \Leftrightarrow SA(T) \in \mathcal{V}^a \Leftrightarrow Tr(SA(T)) \in M \Leftrightarrow SM(T) \in M$$

holds for every tree language T by Remarks 11 and 1, which proves that $\mathcal{V} = M^*$. So, all we have to show is that $\mathcal{A} \in \mathcal{V}^a$ iff $\mathcal{A}^e \in \mathcal{V}^a$ for any \mathcal{A} .

Let $\mathcal{A} = (A, \Sigma)$ be a finite algebra in \mathcal{V}^a . By Lemma 20, any tree language $L \subseteq T(\Lambda_{\mathcal{A}}, X)$ recognized by \mathcal{A}^e can be written as $L = T\varphi^{-1}$, where $\varphi : T(\Lambda_{\mathcal{A}}, X) \rightarrow T(\Sigma, X \cup A)$ is a regular tree homomorphism, and T is a tree language recognized by some power \mathcal{A}^n of \mathcal{A} . Then $\mathcal{A}^n \in \mathcal{V}^a$ implies that $T \in \mathcal{V}(\Sigma, X \cup A)$, and hence $L = T\varphi^{-1} \in \mathcal{V}(\Lambda_{\mathcal{A}}, X)$. This holds for every tree language L recognizable by \mathcal{A}^e , so $\mathcal{A}^e \in \mathcal{V}^a$ by Remark 23.

Now, suppose $\mathcal{A}^e \in \mathcal{V}^a$ for a finite algebra $\mathcal{A} = (A, \Sigma)$. Let $T \subseteq T(\Sigma, X)$ be a tree language recognizable by \mathcal{A} . By Lemma 21, there is a unary ranked alphabet Λ and a regular tree homomorphism $\varphi : T(\Lambda, X \cup \Sigma_0) \rightarrow T(\Sigma, X)$ full with respect to T such that for every $z \in X \cup \Sigma_0$, $L_z = T\varphi^{-1} \cap T(\Lambda, \{z\})$ can be recognized by \mathcal{A}^e as a subset of $T(\Lambda, \{z\})$. So, $L_z \in \mathcal{V}(\Lambda, \{z\})$, thus $L_z \in \mathcal{V}(\Lambda, X \cup \Sigma_0)$. Hence $T\varphi^{-1} = \bigcup_{z \in X \cup \Sigma_0} L_z \in \mathcal{V}(\Lambda, X \cup \Sigma_0)$. Since φ is full with respect to T , then $T \in \mathcal{V}(\Sigma, X)$. This holds for every tree language T recognizable by \mathcal{A} , hence $\mathcal{A} \in \mathcal{V}^a$ by Remark 23. \square

Example 25. It was shown in Example 13 that Def_1 is not definable by syntactic monoids. Here we show that it does not satisfy condition (2) of Theorem 24. Let Σ, X, T, T' be as in Example 13. Define the regular tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Sigma, X)$, by $\varphi_X(x) = x$, $\varphi_X(y) = y$, and $\varphi_2(f) = f(x, f(\xi_1, \xi_2))$, $\varphi_2(g) = g(y, g(\xi_1, \xi_2))$. Now φ is full with respect to T' since for any $t \in T(\Sigma, X)$, if $t \in T'$ then $f(y, x)\varphi\theta_{T'}t$, and if $t \notin T'$ then $g(y, x)\varphi\theta_{T'}t$. Similarly, for $P \in C(\Sigma, X)$, if the leftmost leaf of P is x then $f(y, \xi)\varphi_*\mu_{T'}P$, if the leftmost leaf of P is y then $g(y, \xi)\varphi_*\mu_{T'}P$, and if the leftmost leaf of P is ξ then $\xi\varphi_*\mu_{T'}P$. Clearly $T'\varphi^{-1} = T$, since for any $t \in T(\Sigma, X)$, the leftmost leaf of $t\varphi$ is x iff either $t = x$ or the root of t is f . By Example 13, $T'\varphi^{-1} = T \in \text{Def}_1$, but $T' \notin \text{Def}_1$.

Example 26. Let $\text{Ap} = \{\text{Ap}(\Sigma, X)\}$ be the family of aperiodic tree languages. It was shown to be a GVTL in Example 7.8 of [18]. It is also known that Ap is definable by the variety of aperiodic (syntactic) monoids, see [20]. The argument of Example 7.8 in [18] showing that Ap is closed under inverse g-morphisms can be applied to show that Ap is in fact closed under inverse regular tree homomorphisms. It is also straightforward to see that Ap satisfies condition (1) of Theorem 24. We show that it also satisfies condition (2). Suppose $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is a regular tree homomorphism full with respect to $T \subseteq T(\Omega, Y)$, and $T\varphi^{-1} \in \text{Ap}(\Sigma, X)$. There is an n such that for all $t \in T(\Sigma, X)$ and all $P, Q \in C(\Sigma, X)$, $t \cdot P^n \cdot Q \in T\varphi^{-1} \Leftrightarrow t \cdot P^{n+1} \cdot Q \in T\varphi^{-1}$. For any $s \in T(\Omega, Y)$ and any $R, U \in C(\Omega, Y)$, there are $t \in T(\Sigma, X)$ and $P, Q \in C(\Sigma, X)$ such that $t\varphi\theta_T s$, $P\varphi_*\mu_T R$, and $Q\varphi_*\mu_T U$. So, $s \cdot R^n \cdot U \in T \Leftrightarrow t\varphi \cdot P^n\varphi_* \cdot Q\varphi_* \in T \Leftrightarrow t \cdot P^n \cdot Q \in T\varphi^{-1} \Leftrightarrow t \cdot P^{n+1} \cdot Q \in T\varphi^{-1} \Leftrightarrow t\varphi \cdot P^{n+1}\varphi_* \cdot Q\varphi_* \in T \Leftrightarrow s \cdot R^{n+1} \cdot U \in T$, which shows that $T \in \text{Ap}(\Omega, Y)$.

Example 27. The family of nilpotent tree languages $\text{Nil} = \{\text{Nil}(\Sigma, X)\}$ which consists of finite and cofinite tree languages is a GVFA (see [18], Example 7.5). Let $\Lambda = \Lambda_1 = \{\alpha\}$ be a unary ranked alphabet and $X = \{x, y\}$ be a leaf alphabet. Let $T = \{\alpha(y), \alpha(\alpha(y)), \alpha(\alpha(\alpha(y))), \dots\}$. Clearly $T \in \text{Nil}(\Lambda, \{y\})$, but $T \notin \text{Nil}(\Lambda, X)$.

Hence, Nil does not satisfy the condition (1) of Theorem 24, so it is not definable by syntactic monoids.

5 Definability by Semigroups

In this section, we show how to modify the above results as to yield characterizations of varieties of finite algebras definable by translation semigroups and of varieties of tree languages definable by syntactic semigroups.

5.1 Algebras Definable by Translation Semigroups

The difference between the translation monoid and the translation semigroup of an algebra is that the latter does not automatically contain the identity translation, although it may be included as an elementary translation or as a composition of some elementary translations.

Denote the translation semigroup of an algebra $\mathcal{A} = (A, \Sigma)$ by $\text{TrS}(\mathcal{A})$ and let $\Lambda_{\mathcal{A}}$ be as in Definition 4 except that $\text{Tr}(\mathcal{A})$ is replaced with $\text{TrS}(\mathcal{A})$. We associate with \mathcal{A} a new symbol $\mathbf{I}_{\mathcal{A}}$ that does not appear in $A \cup \Sigma \cup \text{TrS}(\mathcal{A})$. Define the $\Lambda_{\mathcal{A}}$ -algebra $\mathcal{A}^{\varsigma} = (\text{TrS}(\mathcal{A}) \cup \{\mathbf{I}_{\mathcal{A}}\}, \Lambda_{\mathcal{A}})$ by $\bar{p}^{\mathcal{A}^{\varsigma}}(q) = q \cdot p$ and $\bar{p}^{\mathcal{A}^{\varsigma}}(\mathbf{I}_{\mathcal{A}}) = p$ for all $p, q \in \text{TrS}(\mathcal{A})$.

Lemma 28. For any finite algebras $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Omega)$,

- (1) $\text{TrS}(\mathcal{A}) \cong \text{TrS}(\mathcal{A}^{\varsigma})$;
- (2) If $\text{TrS}(\mathcal{A}) \prec \text{TrS}(\mathcal{B})$, then $\mathcal{A}^{\varsigma} \prec_g \mathcal{B}^{\varsigma}$; and
- (3) $\text{TrS}(\mathcal{A}) \times \text{TrS}(\mathcal{B}) \cong \text{Tr}(\kappa(\mathcal{A}^{\varsigma}, \mathcal{B}^{\varsigma}))$ for some g-product $\kappa(\mathcal{A}^{\varsigma}, \mathcal{B}^{\varsigma})$.

Moreover, for any $k \geq 1$, and algebras $\mathcal{A}_1, \dots, \mathcal{A}_k$, there is a g-product \mathcal{P} of $\mathcal{A}_1^{\varsigma}, \dots, \mathcal{A}_k^{\varsigma}$ such that $\text{TrS}(\mathcal{A}_1) \times \dots \times \text{TrS}(\mathcal{A}_k) \cong \text{TrS}(\mathcal{P})$.

Proof. The statements (1) and (3) can be proved similarly as their counterparts in Lemmas 5, 6, and 7 just by replacing the identity translation 1_A (and 1_B) with $\mathbf{I}_{\mathcal{A}}$ (with $\mathbf{I}_{\mathcal{B}}$). We prove (2):

For a semigroup S that satisfies $\text{TrS}(\mathcal{A}) \leftarrow S \subseteq \text{TrS}(\mathcal{B})$, let $\Lambda_S = \{\bar{p} \in \Lambda_{\mathcal{B}} \mid p \in S\}$. Then clearly $S = (S \cup \{\mathbf{I}_{\mathcal{B}}\}, \Lambda_S) \subseteq_g \mathcal{B}^{\varsigma}$ where the interpretation of $\bar{p} \in \Lambda_S$ in S is defined by $\bar{p}^S(q) = q \cdot p$ and $\bar{p}^S(\mathbf{I}_{\mathcal{B}}) = p$ for $p, q \in S$. Suppose $\varphi : S \rightarrow \text{TrS}(\mathcal{A})$ is a semigroup epimorphism. Define the assignment $\kappa : \Lambda_S \rightarrow \Lambda_{\mathcal{A}}$ by $\bar{q}\kappa = \bar{q}\varphi$ for all $q \in S$. It is clear that κ is surjective and for all $q, r \in S \subseteq \text{TrS}(\mathcal{B})$, $(\bar{q}^{\mathcal{B}^{\varsigma}}(r))\varphi = (r \cdot q)\varphi = r\varphi \cdot q\varphi = \bar{q}\varphi^{\mathcal{A}^{\varsigma}}(r\varphi) = (q\kappa)^{\mathcal{A}^{\varsigma}}(r\varphi)$. Hence $(\kappa, \tilde{\varphi}) : S \rightarrow \mathcal{A}^{\varsigma}$ defined by $s\tilde{\varphi} = s\varphi$ for $s \in S$ and $\mathbf{I}_{\mathcal{B}}\tilde{\varphi} = \mathbf{I}_{\mathcal{A}}$, is a g-epimorphism. Thus $\mathcal{A}^e \leftarrow_g S \subseteq_g \mathcal{B}^e$. \square

The following characterization of the class of finite algebras definable by translation semigroups can be proved similarly as Theorem 8.

Theorem 29. A class of finite algebras \mathbf{K} is definable by translation semigroups iff it is a GVFA such that the equivalence $\mathcal{A} \in \mathbf{K}$ iff $\mathcal{A}^{\varsigma} \in \mathbf{K}$ holds for any finite algebra \mathcal{A} .

5.2 Tree Languages Definable by Syntactic Semigroups

Let X be a leaf alphabet and Σ be a ranked alphabet such that $\Sigma \neq \Sigma_0$. A *trivial tree language* T consists of constant or leaf symbols only, i.e., $T \subseteq \Sigma_0 \cup X$. For such a tree language T , the syntactic semigroup of T is the trivial semigroup consisting of a zero element, while its syntactic monoid consists of a zero element and an identity element. Since the trivial semigroup belongs to every variety of finite semigroups, any family of tree languages definable by syntactic semigroups should contain all these trivial tree languages. So, it is reasonable to consider $+$ -varieties of tree languages (cf. [4] Section 11).

The sets of *non-trivial ΣX -trees* and *non-trivial ΣX -contexts* are defined by $T^+(\Sigma, X) = T(\Sigma, X) \setminus (\Sigma_0 \cup X)$ and $C^+(\Sigma, X) = C(\Sigma, X) \setminus \{\xi\}$, respectively. Any subset of $T^+(\Sigma, X)$ is called a *trivial-free tree language*.

For a trivial-free tree language $T \subseteq T^+(\Sigma, X)$ the syntactic semigroup of T is the quotient semigroup $C^+(\Sigma, X)/\mu_T$ where μ_T is restricted to $C^+(\Sigma, X)$.

A regular tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ is called *strict*, if $\varphi_m(f)$ is not trivial for any $f \in \Sigma_m$ with $m > 0$, and $\varphi_X(X), \varphi_0(\Sigma_0) \subseteq Y \cup \Omega_0$ (cf. Definition 11.1 of [4]). We note that if φ is strict and regular, then $T^+(\Sigma, X)\varphi^{-1} = T^+(\Omega, Y)$. A family of regular trivial-free tree languages $\{\mathcal{V}(\Sigma, X)\} \subseteq \{T^+(\Sigma, X)\}$ is called a *$+$ -GVTL* if it is closed under Boolean operations, inverse translations and inverse strict regular tree homomorphisms, and moreover satisfies the following conditions:

(1) For every unary ranked alphabet Λ , and any leaf alphabets X and Y , if $Y \subseteq X$ then $\mathcal{V}(\Lambda, Y) \subseteq \mathcal{V}(\Lambda, X)$.

(2) For any strict regular tree homomorphism $\varphi : T(\Sigma, X) \rightarrow T(\Omega, Y)$ full with respect to $T \subseteq T^+(\Omega, Y)$, if $T\varphi^{-1} \in \mathcal{V}(\Sigma, X)$ then $T \in \mathcal{V}(\Omega, Y)$.

That any variety of trivial-free tree languages definable by syntactic semigroups is a $+$ -GVTL can be proved analogously to that of the monoid case. We claim the converse in the following theorem.

Theorem 30. A family of trivial-free tree languages is definable by syntactic semigroups iff it is a $+$ -GVTL of tree languages.

The proof, once we have proved the following semigroup counterparts of Lemmas 20 and 21, is very similar to that of Theorem 24.

Lemma 31. Let $\mathcal{A} = (A, \Sigma)$ be a finite algebra, and X be a leaf alphabet disjoint from $A \cup \Sigma$.

(1) For any trivial-free tree language $L \subseteq T^+(\Lambda_{\mathcal{A}}, X)$ recognized by \mathcal{A}^c , there exists a strict regular tree homomorphism $\varphi : T(\Lambda_{\mathcal{A}}, X) \rightarrow T(\Sigma, X \cup A)$, and a trivial-free tree language $T \subseteq T^+(\Sigma, X \cup A)$, such that $L = T\varphi^{-1}$, and T can be recognized by a finite power of \mathcal{A} .

(2) For any trivial-free tree language $T \subseteq T^+(\Sigma, X)$ recognized by \mathcal{A} there exists a unary ranked alphabet Λ and a strict regular tree homomorphism $\varphi : T(\Lambda, X \cup \Sigma_0) \rightarrow T(\Sigma, X)$ such that φ is full with respect to T , and for every $z \in X \cup \Sigma_0$, $T\varphi^{-1} \cap T(\Lambda, \{z\})$ can be recognized by \mathcal{A}^c as a subset of $T(\Lambda, \{z\})$.

Proof. (1) Suppose for an initial assignment $\alpha : X \rightarrow \text{Tr}(\mathcal{A}) \cup \{\mathbf{I}_{\mathcal{A}}\}$ and a subset $F \subseteq \text{Tr}(\mathcal{A}) \cup \{\mathbf{I}_{\mathcal{A}}\}$, $L = \{t \in \text{T}(\Lambda_{\mathcal{A}}, X) \mid t\alpha^{A^0} \in F\}$ holds. Since L is trivial-free, we can assume that $\mathbf{I}_{\mathcal{A}} \notin F$, or equivalently $F \subseteq \text{Tr}(\mathcal{A})$. Let $Y = \{x \in X \mid x\alpha = \mathbf{I}_{\mathcal{A}}\}$. Define the tree homomorphism $\varphi : \text{T}(\Lambda_{\mathcal{A}}, X) \rightarrow \text{T}(\Sigma, X \cup A)$ by $\varphi_X(x) = x$ for all $x \in X$, and for every $p \in \text{Tr}(\mathcal{A})$ choose a $\varphi_1(\bar{p}) \in C(\Sigma, A)$ such that $\varphi_1(\bar{p})^A = p$. Obviously φ is a strict regular tree homomorphism. Suppose that $A = \{a_1, \dots, a_m\}$. Let $F' = \{(p(a_1), \dots, p(a_m)) \in A^m \mid p \in F\}$, and define the initial assignment $\beta : X \cup A \rightarrow A^m$ by $x\beta = ((x\alpha)(a_1), \dots, (x\alpha)(a_m))$ for all $x \in X \setminus Y$, $y\beta = (a_1, \dots, a_m)$ for all $y \in Y$, and $a\beta = (a, \dots, a) \in A^m$ for all $a \in A$. Let T be the subset of $\text{T}(\Sigma, X \cup A)$ recognized by (A^m, β, F') . We show $L = T\varphi^{-1}$. Every trivial-free tree w in $\text{T}^+(\Lambda_{\mathcal{A}}, X)$ is of the form $w = \bar{p}_1(\bar{p}_2(\dots \bar{p}_k(x)\dots))$ for some $p_1, \dots, p_k \in \text{Tr}(\mathcal{A})$ ($k > 0$) and $x \in X$. For such a tree w , $w\alpha^{A^0} = x\alpha \cdot p_k \dots p_2 \cdot p_1$ if $x \in X \setminus Y$, and $w\alpha^{A^0} = p_k \dots p_2 \cdot p_1$ if $x \in Y$; also $(w\varphi)\beta^{A^m} = (x\alpha \cdot p_k \dots p_2 \cdot p_1(a_1), \dots, x\alpha \cdot p_k \dots p_2 \cdot p_1(a_m))$ holds. So, for $x \in X \setminus Y$ we have $w\varphi \in T$ iff $(w\varphi)\beta^{A^m} \in F'$ iff “for some $p \in F$, $p(a) = x\alpha \cdot p_k \dots p_2 \cdot p_1(a)$, for all $a \in A$ ” iff $x\alpha \cdot p_k \dots p_2 \cdot p_1 \in F$ iff $w\alpha^{A^0} \in F$ iff $w \in L$. Similarly, for $x \in Y$ we have $w\varphi \in T$ iff $(w\varphi)\beta^{A^m} \in F'$ iff “for some $p \in F$, $p(a) = p_k \dots p_2 \cdot p_1(a)$, for all $a \in A$ ” iff $p_k \dots p_2 \cdot p_1 \in F$ iff $w\alpha^{A^0} \in F$ iff $w \in L$.

(2) The proof is almost identical to that of Lemma 21, only $1_{\mathcal{A}}$ is replaced with $\mathbf{I}_{\mathcal{A}}$. \square

It was shown in Example 13 that the variety of 1-definite tree languages is not definable by syntactic monoids. In the following example we show that its trivial-free counterpart is not definable by syntactic semigroups.

Example 32. The syntactic semigroup of the trivial-free 1-definite tree language $T \setminus \{x\}$ where T is defined in Example 13, consists of two elements both of which are right zeros. Let $\Lambda = \Lambda_1 = \{\alpha, \beta\}$ and $X = \{x, y\}$. Let T'' be the set of all ΛX -trees which either have root label α and leaf label x or have root label β and leaf label y , i.e., $T'' = \{\alpha(p(x)) \mid p \in C(\Lambda, X)\} \cup \{\beta(p(y)) \mid p \in C(\Lambda, X)\}$. It is easy to see that the syntactic semigroup of T'' consists of two right zero elements, but clearly T'' is not 1-definite. So, the trivial-free 1-definite tree languages are not definable by syntactic semigroups.

Indeed, T'' is not k -definite for any $k \geq 1$, thus the trivial-free definite tree languages are not definable by syntactic semigroups.

5.3 Monoids vs. Semigroups

In this subsection we show that the concepts of “definability by semigroups” and “definability by monoids” are not comparable to each other.

The abbreviation VFS stands for variety of finite semigroups. For a VFS \mathbf{S} , let \mathbf{S}^a be the class of all finite algebras whose translation semigroups are in \mathbf{S} , and \mathbf{S}^t be the family of all recognizable trivial-free tree languages whose syntactic semigroups are in \mathbf{S} (cf. Definitions 3 and 12).

We recall Proposition 10.9 of [18] which can be extended to VFS's.

Theorem 33. For any VFM M and VFS S , the identities $M^{at} = M^t$, $M^{ta} = M^a$, $S^{at} = S^t$ and $S^{ta} = S^a$ hold.

Theorem 34. (1) There is a VFM M for which no VFS S , satisfies $M^a = S^a$ or $M^t = S^t$.

(2) There is a VFS S such that for no VFM M , $M^a = S^a$ or $M^t = S^t$ holds.

Proof. (1) Let M be the class of all finite monoids which satisfy the equation $y \cdot x \cdot x = y$. Obviously, M is a VFM. Let $\Sigma = \Sigma_1 = \{f\}$ and put the algebras $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Sigma)$ be defined by $A = \{a\}$, $f^{\mathcal{A}}(a) = a$, and $B = \{a, b\}$, $f^{\mathcal{B}}(a) = f^{\mathcal{B}} = a$. Then $\text{Tr}(\mathcal{A}) \cong \text{TrS}(\mathcal{A}) \cong \text{TrS}(\mathcal{B})$ is the trivial semigroup, but the monoid $\text{Tr}(\mathcal{B})$ consists of a zero element (0) and a unit (1). Now, $\mathcal{A} \in M^a$, but $\mathcal{B} \notin M^a$ since $\text{Tr}(\mathcal{B})$ does not satisfy the equation $y \cdot x \cdot x = y$: $1 \cdot 0 \cdot 0 = 0 \neq 1$. Hence, M^a is not definable by translation semigroups. Now if $M^t = S^t$ hold for a VFS S , then by Theorem 33 we would have $M^a = M^{ta} = S^{ta} = S^a$, contradiction.

(2) Let S be the variety of finite right zero semigroups, i.e., the class of all semigroups that satisfy the equation $y \cdot x = x$. It can be easily seen that if T and T' are the tree languages of Example 13, then $T \setminus \{x\} \in S^t(\Sigma, X)$ since the syntactic semigroup of $T \setminus \{x\}$ has two elements both of which are right zeros. On the other hand, the syntactic semigroup of T' consists of an identity element and two right zeros (like its syntactic monoid). Thus $T' \notin S^t(\Sigma, X)$. This shows that S^t is not definable by syntactic monoids (since $T \setminus \{x\}$ and T' have isomorphic syntactic monoids) whence $M^t = S^t$ does not hold for any VFM M . On the other hand, if $M^a = S^a$ holds for some VFM M , then by Theorem 33, $M^t = M^{at} = S^{at} = S^t$, contradiction. \square

Theorems 34 justifies the task of studying the definability by semigroup separately from the monoid case.

6 String languages definable by translation monoids

In this final section, we present for strings the results corresponding to those of the previous sections. Familiarity with the basic notions of string languages and automata are presumed.

Let X be a finite alphabet, and X^* be the set of words over X . A string language over X is any subset of X^* . In the literature the syntactic monoid $\text{SM}(L)$ of a string language $L \subseteq X^*$ is defined to be the quotient monoid X^*/θ_L where $w\theta_L w' \iff \forall u, v \in X^* (uwv \in L \iff ww'v \in L)$.

For a monoid $\mathcal{M} = (M, \cdot)$ the translations of \mathcal{M} are the unary functions on M defined by $x \mapsto m \cdot x \cdot m'$ for some $m, m' \in M$. Denote the composition of the translations p and q by $p \circ q$, that is $p \circ q(m) = p(q(m))$ for all $m \in M$. We note that the set of translations of \mathcal{M} is a monoid with respect to composition operation.

Denote the translation monoid of \mathcal{M} by $\text{Tr}(\mathcal{M})$. For a string language L , let the *translation monoid* $\text{TM}(L)$ of L be the translation monoid of the syntactic monoid of L , i.e., $\text{TM}(L) = \text{Tr}(\text{SM}(L))$.

Note that by necessity the terms ‘syntactic monoid’ and ‘translation monoid’ have different meanings and interpretations in this section.

Eilenberg’s [3] variety theorem establishes a correspondence between a variety of finite monoids \mathbf{M} and a variety of string languages $\mathcal{L} = \{\mathcal{L}(X)\}$ such that for any $L \subseteq X^*$, $L \in \mathcal{L}(X) \iff \text{SM}(L) \in \mathbf{M}$.

A variety of string languages $\mathcal{V} = \{\mathcal{V}(X)\}$ is definable by translation monoids if there exists a variety of finite monoids \mathbf{M} such that for any $L \subseteq X^*$, $L \in \mathcal{V}(X) \iff \text{TM}(L) \in \mathbf{M}$. We shall characterize these varieties of string languages in Theorem 40 below.

It is known that not any variety of string languages can be defined by translation monoids (one example is the class of reverse definite, or frontier testable, string languages, cf. [21]).

For a monoid $\mathcal{M} = (M, \cdot)$, the *reverse* of \mathcal{M} is the monoid $\mathcal{M}^R = (M, \cdot_R)$ where $m \cdot_R m' = m' \cdot m$ for $m, m' \in M$. Clearly $(\mathcal{M}^R)^R = \mathcal{M}$. We show that a variety of finite monoids is definable by translation monoids (see Definition 3) iff it is closed under the reversing operation.

First, we show that the reverse of a monoid and the original monoid have isomorphic translation monoids.

Lemma 35. For a monoid $\mathcal{M} = (M, \cdot)$, $\text{Tr}(\mathcal{M}) \cong \text{Tr}(\mathcal{M}^R)$.

Proof. For any translation $p(x) = m \cdot x \cdot m'$ ($m, m' \in M$) of \mathcal{M} let $p^R(x) = m' \cdot x \cdot m$. The mapping $\text{Tr}(\mathcal{M}) \rightarrow \text{Tr}(\mathcal{M}^R)$, $p \mapsto p^R$ is an isomorphism. \square

Next, we present some connections between the translation monoid of a monoid and the original monoid.

Lemma 36. For any monoid \mathcal{M} , (1) $\mathcal{M} \subseteq \text{Tr}(\mathcal{M})$, and (2) $\text{Tr}(\mathcal{M}) \leftarrow \mathcal{M} \times \mathcal{M}^R$.

Proof. Let $\mathcal{M} = (M, \cdot)$. (1) For any $m \in M$, let p_m be the translation defined by $p_m(x) = m \cdot x$ on M . It is easy to see that the mapping $m \mapsto p_m$ is a monoid monomorphism that embeds \mathcal{M} into $\text{Tr}(\mathcal{M})$. (2) For any $m, n \in M$, let $q_{(m,n)}$ be the translation of \mathcal{M} defined by $q_{(m,n)}(x) = m \cdot x \cdot n$. It can be easily seen that $(m, n) \mapsto m \cdot x \cdot n$ yields an epimorphism $\mathcal{M} \times \mathcal{M}^R \twoheadrightarrow \text{Tr}(\mathcal{M})$. \square

Finally, we characterize the varieties of finite monoids definable by translation monoids.

Theorem 37. A variety of finite monoids \mathbf{M} is definable by translation monoids iff it is closed under the reversing operation, i.e., $\mathcal{M} \in \mathbf{M} \Rightarrow \mathcal{M}^R \in \mathbf{M}$ for any monoid \mathcal{M} .

Proof. By Lemma 35, every variety of finite monoids definable by translation monoids is closed under the reversing operation. Now suppose a variety of finite monoids \mathbf{M} is closed under the reversing operation. We show that $\mathcal{M} \in \mathbf{M} \iff$

$\text{Tr}(\mathcal{M}) \in \mathbf{M}$ for any monoid \mathcal{M} . The implication $\text{Tr}(\mathcal{M}) \in \mathbf{M} \Rightarrow \mathcal{M} \in \mathbf{M}$ follows from Lemma 36(1). For the converse, let $\mathcal{M} \in \mathbf{M}$. Then also $\mathcal{M}^R \in \mathbf{M}$, and hence $\text{Tr}(\mathcal{M}) \in \mathbf{M}$ by Lemma 36(2). \square

The proof also implies that:

Corollary 38. If a variety of finite monoids \mathbf{M} is definable by translation monoids, then \mathbf{M} is generated by the translation monoids of its members.

In the sequel we characterize the varieties of string languages definable by translation monoids.

For a string $w = x_1x_2 \dots x_n \in X^*$ define the reverse of w as $w^R = x_n \dots x_2x_1$. We note that $u^Rv^R = (vu)^R$ holds for all $u, v \in X^*$. For a string language $L \subseteq X^*$, $L^R = \{w^R \in X^* \mid w \in L\}$.

The following lemma is a known fact (see e.g. [3]).

Lemma 39. For any string language $L \subseteq X^*$, $\text{SM}(L^R) \cong \text{SM}(L)^R$.

Our characterization of the varieties of string languages definable by translation monoids is the following.

Theorem 40. A class of string languages \mathcal{V} is definable by translation monoids iff it is a variety of string languages closed under the reversing operation, i.e., $L \in \mathcal{V}(X) \Rightarrow L^R \in \mathcal{V}(X)$ for any string language $L \subseteq X^*$.

Proof. Since Lemmas 39 and 35 imply that $\text{TM}(L) \cong \text{TM}(L^R)$ for any string language L , any variety of string languages definable by translation monoids is closed under the reversing operation. Now, suppose \mathcal{V} is a variety of string languages closed under the reversing operation. By Eilenberger's variety theorem there is a variety of finite monoids \mathbf{M} such that for any string language $L \subseteq X^*$, $L \in \mathcal{V}(X) \Leftrightarrow \text{SM}(L) \in \mathbf{M}$. We show that the class \mathbf{M} also defines the translation monoids of \mathcal{V} , that is to say, for any $L \subseteq X^*$, $L \in \mathcal{V}(X) \Leftrightarrow \text{TM}(L) \in \mathbf{M}$. First, suppose L is in $\mathcal{V}(X)$. Then also $L^R \in \mathcal{V}(X)$, so $\text{SM}(L) \in \mathbf{M}$ and $\text{SM}(L^R) \in \mathbf{M}$. By Lemma 39, $\text{SM}(L)^R \in \mathbf{M}$, and since $\text{TM}(L)$ is an epimorphic image of $\text{SM}(L) \times \text{SM}(L)^R$ by Lemma 36, $\text{TM}(L) \in \mathbf{M}$. Next, suppose $\text{TM}(L) \in \mathbf{M}$ for a string language $L \subseteq X^*$. Since by Lemma 36, $\text{SM}(L)$ is isomorphic to a submonoid of $\text{TM}(L)$, then $\text{SM}(L) \in \mathbf{M}$, and hence $L \in \mathcal{V}(X)$. \square

Corollary 41. Let \mathcal{V} be a variety of string languages definable by translation monoids. Then the variety generated by the translation monoids of \mathcal{V} is equal to the variety generated by the syntactic monoids of \mathcal{V} .

An analogue of Theorem 40 can be proved for translation semigroups.

Unlike Theorems 24 and 30 for tree languages, by Theorem 40 checking whether or not a variety of string languages is definable by translation monoids or semigroups is rather easy. For example the variety of definite string languages and the variety of reverse definite string languages are not definable by translation semigroups, while the variety of aperiodic string languages and the variety of commutative string languages (i.e., having commutative syntactic monoids) are definable by translation monoids.

Acknowledgements

I would like to thank Magnus Steinby, Tatjana Petković, Ville Piirainen, and Zoltán Ésik for helpful comments and stimulating discussions.

References

- [1] Almeida J., On pseudovarieties, varieties of languages, filters of congruences, pseudoidentities and related topics, *Algebra Universalis* **27** (1990) 333–350.
- [2] Birkhoff G., On the structure of abstract algebras, *Proc. Cambridge Phil. Soc.* **31** (1935) 433–454.
- [3] Eilenberg S., *Automata, Languages, and Machines*, Vol. **B**. Pure and Applied Mathematics, Vol. 59, Academic Press, New York – London (1976).
- [4] Ésik Z., A variety theorem for trees and theories, Automata and formal languages VIII (Salgótarján, 1996), *Publ. Math. Debrecen* **54** (1999), suppl., 711–762.
- [5] Ésik Z. & Weil P., On Logically Defined Recognizable Tree Languages, *Proceedings of FSTTCS'03*, Lect. Notes in Comp. Sci. **2914**, Springer-Verlag (2003), 195–207.
- [6] Nivat M. & Podelski A., Tree monoids and recognizability of sets of finite trees, *Resolution of Equations in Algebraic Structures*, Vol. 1, Academic Press, Boston MA (1989) 351–367.
- [7] Nivat M. & Podelski A., Definite tree languages (cont'd), *Bull. EATCS* **38** (1989) 186–190.
- [8] Petković T., Ćirić M. & Bogdanović S., Unary algebras, semigroups and congruences on free semigroups, *Theoret. Comput. Sci.* **324** (2004) 87–105.
- [9] Petković T., Ćirić M. & Bogdanović S., Eilenberg type theorems for automata, submitted.
- [10] Pin J.E., *Varieties of formal languages*, Foundations of Computer Science, Plenum Publishing Corp., New York (1986).
- [11] Pin J.E., A variety theorem without complementation, *Izvestiya VUZ Matematika* **39** (1995) 80–90.
- [12] Piirainen V., Monotone algebras, R -trivial monoids and a variety of tree languages, *Bull. EATCS* **84** (2004) 189–194.
- [13] Podelski A., A monoid approach to tree languages, in: Nivat M. & Podelski A. (ed.) *Tree Automata and Languages*, Elsevier-Amsterdam (1992) 41–56.

- [14] Salomaa K., *Syntactic monoids of regular forests* (Finnish), M.Sc. Thesis, Department of Mathematics, Turku University (1983).
- [15] Schützenberger M. P., On finite monoids having only trivial subgroups, *Information and Control* **8** (1965) 190–194.
- [16] Steinby M., Syntactic algebras and varieties of recognizable sets, in: *Proc. CAAP'79* (University of Lille 1979) 226–240.
- [17] Steinby M., A theory of tree language varieties, in: Nivat M. & Podelski A. (ed.) *Tree Automata and Languages*, Elsevier-Amsterdam (1992) 57–81.
- [18] Steinby M., General varieties of tree languages, *Theoret. Comput. Sci.* **205** (1998) 1–43.
- [19] Thérien D., Recognizable languages and congruences, *Semigroup Forum* **23** (1981) 371–373.
- [20] Thomas W., Logical aspects in the study of tree languages, *Ninth Colloquium on Trees in Algebra and in Programming* (Proc. CAAP'84), Cambridge University Press (1984) 31–51.
- [21] Wilke T., An algebraic characterization of frontier testable tree languages, *Theoret. Comput. Sci.* **154** (1996) 85–106.

Received August, 2004

Topologies for the Set of Disjunctive ω -words

Ludwig Staiger*

Abstract

An infinite sequence (ω -word) is referred to as disjunctive provided it contains every finite word as infix (factor). As Jürgensen and Thierrin [JT83] observed the set of disjunctive ω -words, D , has a trivial syntactic monoid but is not accepted by a finite automaton.

In this paper we derive some topological properties of the set of disjunctive ω -words. We introduce two non-standard topologies on the set of all ω -words and show that D fulfills some special properties with respect to these topologies:

In the first topology – the so-called topology of forbidden words – D is the smallest nonempty G_δ -set, and in the second one D is the set of accumulation points of the whole space as well as of itself.

In 1983 two papers dealing with the ω -language of disjunctive ω -words appeared [JST83, JT83]. In the latter it was shown that this ω -language is a natural example of an ω -language having a trivial (finite) syntactic monoid but not being accepted by a finite automaton. For a more detailed account see [St83, JT86].

Subsequently, disjunctive ω -words became of interest in connection with random and Borel normal sequences (see, for instance, [Ca02, He96]). In contrast to Borel normality, “disjunctivity” is a natural qualitative property which is satisfied, in particular, by Borel normal and by random ω -words.

As in [JST83, JT83] we say that an ω -word is *disjunctive* if it contains any (finite) word as a subword. In this paper we are going to investigate topological properties of the set of all disjunctive sequences (ω -words). Usually, one considers the space of all ω -words over a finite alphabet X as the infinite product space of the discrete space X . Introducing the Baire metric, this space can be considered as a metric space (Cantor space) (X^ω, ρ) , that is, a compact totally disconnected space.

In this paper we consider topologies on the set of all ω -words over a finite alphabet X in which the set of all disjunctive ω -words has a special property:

First, we consider the topology of “forbidden words” in which the set of disjunctive ω -words is the smallest G_δ -set. The second topology is a special case of the topologies derived from formal languages (cf. [St87]). Here the set of disjunctive ω -words turns out to be the largest set which is closed and dense in itself.

*Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg, von-Seckendorff-Platz 1, D-06099 Halle, Germany. E-mail: staiger@informatik.uni-halle.de

1 Notation

By $\mathbb{N} = \{0, 1, 2, \dots\}$ we denote the set of natural numbers. Let X be our alphabet of cardinality $\#X = r$, $r \in \mathbb{N}$, $r \geq 2$.

By X^* we denote the set of finite strings (words) on X , including the *empty* word e . We consider the space X^ω of infinite sequences (ω -words) over X . For $w \in X^*$ and $\eta \in X^* \cup X^\omega$ let $w \cdot \eta$ be their *concatenation*. This concatenation product extends in an obvious way to subsets $W \subseteq X^*$ and $B \subseteq X^* \cup X^\omega$.

We extend the operations $*$ and $^\omega$ to arbitrary subsets $W \subseteq X^*$ in the usual way :

$$\begin{aligned} W^* &:= \bigcup_{n \in \mathbb{N}} W^n \text{ where } W^0 := \{e\}, W^{n+1} := W^n \cdot W, \text{ and} \\ W^\omega &:= \{w_0 \cdot w_1 \cdot \dots \cdot w_i \cdot \dots : i \in \mathbb{N} \wedge w_i \in W \setminus \{e\}\} \end{aligned}$$

is the set of ω -words in X^ω formed by concatenating members of W .

We will refer to subsets of X^* and X^ω as languages or ω -languages, respectively.

By " \sqsubseteq " we denote the prefix relation, that is, $w \sqsubseteq \eta$ if and only if there is an η' such that $w \cdot \eta' = \eta$, and $\mathbf{A}(\eta) := \{w : w \in X^* \wedge w \sqsubseteq \eta\}$ and $\mathbf{A}(B) := \bigcup_{\eta \in B} \mathbf{A}(\eta)$ are the languages of finite prefixes of η and B , respectively.

The set of subwords (infixes) of $\eta \in X^* \cup X^\omega$ will be denoted by $\mathbf{T}(\eta) := \{w : w \in X^* \wedge \exists v(vw \sqsubseteq \eta)\}$.

An ω -language F is called *regular* provided there is an $n \in \mathbb{N}$ and regular languages W_i, V_i ($1 \leq i \leq n$) such that

$$F = \bigcup_{i=1}^n W_i V_i^\omega. \quad (1)$$

Similarly, an ω -language F is called *context-free* if F has the form of Eq. (1) where W_i and V_i are context-free languages.

Observe, that $V^\omega = \emptyset$, $V^\omega = \{v\}^\omega$ or $V^\omega \supseteq \{v, u\}^\omega$ for some words $v, u \in V^*$ with $|v| = |u| > 0$ and $v \neq u$. Thus, every at most countable context-free ω -language consists entirely of ultimately periodic ω -words (cf. [St97]).

2 Preliminary Considerations

In the study of ω -languages it is useful to consider X^ω as a metric space (Cantor space) with the following metric.

$$\rho(\eta, \xi) := \inf\{r^{-|w|} : w \sqsubset \eta \wedge w \sqsubset \xi\} \quad (2)$$

or an equivalent one¹.

¹For example, the Baire metric $\varrho(\eta, \xi) := \inf\{\frac{1}{1+|w|} : w \sqsubset \eta \wedge w \sqsubset \xi\}$ generates the same topology.

In this paper, however, we will consider also a topology on X^ω which cannot be specified by a metric, that is, a so-called non-metrizable topology. To this end we introduce topologies on X^ω in the general way (cf. [Ku66, En77]).

A *topology* in X^ω is a family $\mathcal{O} \subseteq \mathcal{P}(X^\omega)$ of subsets of X^ω such that $\emptyset, X^\omega \in \mathcal{O}$ and \mathcal{O} is closed under finite intersection and arbitrary union. The sets in \mathcal{O} are called *open* subsets of X^ω . The complements of open subsets are referred to as *closed*. Since an arbitrary intersection of closed sets is again closed, every set $F \subseteq X^\omega$ is contained in a minimal closed set, its *closure* $\mathcal{C}_{\mathcal{O}}(F)$.

Having defined open and closed sets for some topology in X^ω , we proceed to the next classes of the Borel hierarchy (cf. [Ku66]):

\mathbf{G}_δ is the set of countable intersections of open subsets of X^ω ,

\mathbf{F}_σ is the set of countable unions of closed subsets of X^ω .

A metric σ generates the set of open sets \mathcal{O}_σ in the following way: First we define the *open balls* $B_\epsilon(\xi) := \{\eta : \sigma(\xi, \eta) < \epsilon\}$ for $\epsilon > 0$. Then a set is open in the space (X^ω, σ) if it is a union of open balls. In Cantor space, open balls are of the form $w \cdot X^\omega$, and, consequently, the set of open subsets of X^ω is $\mathcal{O}_C = \{W \cdot X^\omega : W \subseteq X^*\}$. From this it follows that a subset $F \subseteq X^\omega$ is closed in Cantor space if and only if $\mathbf{A}(\xi) \subseteq \mathbf{A}(F)$ implies $\xi \in F$, and the closure in Cantor space can be specified as $\mathcal{C}(F) := \{\xi : \mathbf{A}(\xi) \subseteq \mathbf{A}(F)\}$.

In Section 4 we shall consider the so-called topology of “forbidden” words which is specified by the set of open sets $\mathcal{O}_T := \{X^* \cdot W \cdot X^\omega : W \subseteq X^*\}$.²

This topology is a subtopology of Cantor topology $\mathcal{O}_C \supset \mathcal{O}_T$, or, equivalently, the Cantor space is a refinement of the topology of “forbidden” words.

Finally, we define, for a language $W \subseteq X^*$, its δ -*limit* of W , W^δ , which consists of all infinite sequences of X^ω that contain infinitely many prefixes in W ,

$$W^\delta = \{\xi \in X^\omega : \#(\mathbf{A}(\xi) \cap W) = \infty\}.$$

For \mathbf{G}_δ -sets in Cantor space we have the following characterization via languages (cf. [Th90, St87, St97]). It explains also why we call W^δ the δ -limit of the language W .

Theorem 1. *In Cantor space, a subset $F \subseteq X^\omega$ is a \mathbf{G}_δ -set if and only if there is a language $W \subseteq X^*$ such that $F = W^\delta$.*

3 The ω -Language of Disjunctive Sequences

In this section we will present a few simple general properties of the ω -language D of all disjunctive sequences over X , and its topological properties in Cantor space. Some of the results in this section are reported in [CPS97, St02].

²The term *forbidden* refers to the fact that closed subsets are specified by forbidding a certain set W of infixes.

As in [JST83, JT83] an ω -word $\xi \in X^\omega$ is called *disjunctive* provided $\mathbf{T}(\xi) = X^*$. Thus the set of all disjunctive ω -words satisfies $D = \{\xi : \mathbf{T}(\xi) = X^*\}$.

From this definition we obtain

$$D = \bigcap_{w \in X^*} X^* w X^\omega. \quad (3)$$

Our next lemma shows that D is an example of a ω -language which has a trivial finite syntactic congruence but is not context-free. The proof refers to the investigations of Jürgensen and Thierrin [JT83, JT86].

The *syntactic congruence* \sim_F of an ω -language $F \subseteq X^\omega$ is defined as follows³

$$w \sim_F v :\Leftrightarrow \forall u \forall \xi (u \in X^* \wedge \xi \in X^\omega \rightarrow (uw\xi \in F \leftrightarrow uv\xi \in F)).$$

As usual, we call \sim_F of *finite index* iff its number of equivalence classes is finite.

Observe that $\mathbf{T}(uw\xi) = X^*$ iff $\mathbf{T}(\xi) = X^*$. Thus it is clear that $w \sim_D v$ for arbitrary $w, v \in D$, and \sim_D has exactly one equivalence class which coincides with X^* . Thus we have proven the first part of the following.

Lemma 2 ([JT83]). *The ω -language D has a syntactic congruence of finite index but is not context-free.*

Proof. As $\mathbf{T}(\prod_{w \in X^*} w) = X^*$ and $\mathbf{T}(wv^\omega) \neq X^*$, D is nonempty and does not contain an ultimately periodic ω -word wv^ω . Following Eq. (1) the ω -language D cannot be context-free. \square

The representation of Eq. (3) verifies that D is a \mathbf{G}_δ -set in Cantor space. Thus, in view of Theorem 1 it can be represented as the δ -limit of a language. In case of D we construct such a language W_D explicitly (cf. [St02]).

Proposition 3. *Let $W_D = \{wx : w \in X^* \wedge x \in X \wedge \exists n(n \leq |w| + 1 \wedge \mathbf{T}(wx) \supseteq X^n \wedge \mathbf{T}(w) \not\supseteq X^n)\}$. Then $D = W_D^\delta$.*

Finally, we are going to show that the topological complexity of D in Cantor space cannot be decreased. To this end we quote Theorem 21 from [St83].

Theorem 4 ([St83]). *If $F \subseteq X^\omega$ has a syntactic congruence of finite index and is simultaneously an \mathbf{F}_σ - and a \mathbf{G}_δ -set in Cantor space, then F is regular.*

Combining Theorem 4 with Lemma 2 and Eq. (3) we get:

Proposition 5. *In Cantor space, D is not an \mathbf{F}_σ -set.*

³There are other notions of syntactic congruences for ω -languages in use (cf. [MS97]).

4 The Topology of Forbidden Words

In this section we investigate the topology of forbidden words described above and its relation to the set of disjunctive sequences. It turns out that this topology is not a metric one.

Recall $\mathcal{O}_T = \{X^*WX^\omega \mid W \subseteq X^*\}$ from Section 2. As $X^*VX^\omega \cap X^*WX^\omega = (X^*WX^* \cap X^*VX^*)X^\omega$ this family \mathcal{O}_T is closed under finite intersection. The closure under arbitrary union is obvious. Thus it defines a topology on X^ω .

An ω -language $F \subseteq X^\omega$ avoids words of a language $W \subseteq X^*$ provided $F \subseteq X^\omega \setminus X^*WX^\omega$, that is, no word $w \in W$ occurs as a subword (infix) of an ω -word $\xi \in F$. Therefore, the closed sets in the topology \mathcal{O}_T are characterized by the fact that their ω -words do not contain subwords from W . The following theorem gives a connection to closed sets in Cantor space.

To this end we define $F/w := \{\xi : w\xi \in F\}$.

Theorem 6. *Let $F \subseteq X^\omega$. Then the following conditions are equivalent:*

1. *F is closed in the topology of forbidden words.*
2. *F is closed in Cantor space and $\forall w(w \in X^* \Rightarrow F \supseteq F/w)$.*
3. *F is closed in Cantor space and $\mathbf{A}(F) = \mathbf{T}(F)$.*
4. *$\forall \xi(\mathbf{A}(\xi) \subseteq \mathbf{T}(F) \Rightarrow \xi \in F)$.*

Proof. “1. \Rightarrow 2” : As we noticed above, every ω -language closed in the topology of forbidden words is also closed in Cantor’s topology. Let $w \in X^*$ and $F = X^\omega \setminus X^*WX^\omega$. Then $F/w = X^\omega \setminus (X^*WX^\omega)/w$, and the assertion follows from the obvious inclusion $(X^*WX^\omega)/w \supseteq X^*WX^\omega$.

“2. \Rightarrow 3.” follows from the identity $\mathbf{A}(\bigcup_{w \in X^*} F/w) = \mathbf{T}(F)$.

“3. \Rightarrow 4.” : If F is closed in Cantor space we have $F = \{\xi : \mathbf{A}(\xi) \subseteq \mathbf{A}(F)\}$. Now the assertion 4. follows from $\mathbf{A}(F) = \mathbf{T}(F)$.

Finally, we show that Condition 4 implies $F = X^\omega \setminus X^* \cdot (X^* \setminus \mathbf{T}(F)) \cdot X^\omega$. Since $X^* \setminus \mathbf{T}(F) = X^* \cdot (X^* \setminus \mathbf{T}(F)) \cdot X^*$ it suffices to prove that $F = X^\omega \setminus (X^* \setminus \mathbf{T}(F)) \cdot X^\omega$.

The inclusion $F \subseteq X^\omega \setminus (X^* \setminus \mathbf{A}(F)) \cdot X^\omega \subseteq X^\omega \setminus (X^* \setminus \mathbf{T}(F)) \cdot X^\omega$ follows from $\mathbf{A}(F) \subseteq \mathbf{T}(F)$. To prove the converse inclusion let $\xi \notin F$. Then in view of Condition 4 there is a prefix $w \sqsubset \xi$ such that $w \notin \mathbf{T}(F)$. Consequently, $\xi \in (X^* \setminus \mathbf{T}(F)) \cdot X^\omega$. \square

In view of the equivalence “1. \Leftrightarrow 4.” we obtain the following representation of the closure operator \mathcal{C}_T defined by the topology of forbidden words:

$$\mathcal{C}_T(F) = \{\xi : \mathbf{A}(\xi) \subseteq \mathbf{T}(F)\}.$$

Recall that the closure in Cantor space was definable as $\mathcal{C}(F) = \{\xi : \mathbf{A}(\xi) \subseteq \mathbf{A}(F)\}$.

The additional requirements $\forall w(w \in X^* \Rightarrow F \supseteq F/w)$ and $\mathbf{A}(F) = \mathbf{T}(F)$ in 2. and 3. are, however, not equivalent in general. The following example shows that there is an ω -language (necessarily not closed in Cantor space) which satisfies $\mathbf{A}(F) = \mathbf{T}(F)$, but not the condition $\forall w(w \in X^* \Rightarrow F \supseteq F/w)$.

Example 1. Let $F = (X^2)^* bba^\omega \cup X(X^2)^* aab^\omega$. Then $\mathbf{A}(F) = \mathbf{T}(F) = X^*$, but $F/a \not\subseteq F$.

Since the family of regular ω -languages is closed under Boolean operations, the ω -language $F_W = X^\omega \setminus X^* W X^\omega$ is regular if the language of forbidden patterns $W \subseteq X^*$ is regular. In connection with Eq. (1) and the considerations on V^ω immediately following it this yields as a consequence the following generalization of a result of El-Zanati and Transue [ET90].

Theorem 7. Let $W \subseteq X^*$ be a regular language. If F_W is uncountable, then F_W contains a subset of the form $w\{u, v\}^\omega$, where $u \neq v$ and $|u| = |v| > 0$.

We continue with some more examples. The first is an example of a countable regular ω -language F_W which requires an infinite set of forbidden patterns.

Example 2. Let $X = \{a, b\}$ and $W = ba^*b$. Then $F_W = X^\omega \setminus X^* W X^\omega = a^*ba^\omega \cup a^\omega$ is a countable ω -language. It is clear that $F_W \neq F_V$, for any finite language $V \subseteq X^*$.

Though the regularity of W implies the regularity of F_W this same relation is not true for context-free languages and ω -languages.

Example 3. Let $X = \{a, b\}$ and $W = \{bb\} \cup \{ba^i ba^j b \mid j \neq i + 1\}$. Clearly, W is a deterministic context-free language, and $F_W = a^*(\{\eta_i \mid i \in \mathbb{N}\} \cup \{\eta_{i,j} \mid i, j \in \mathbb{N} \wedge i \leq j\})$ where $\eta_i = ba^i ba^{i+1} b \dots$ and $\eta_{i,j} = ba^i ba^{i+1} \dots ba^j ba^\omega$. Since F_W is countable but does not consist entirely of ultimately periodic ω -words, Eq. (1) shows that F_W is not context-free.

Finally, we discuss a characterization of the ω -language of disjunctive sequences D by means of the topology of forbidden words. From Eq. (3) we obtain immediately

Proposition 8. In the topology of forbidden words, D is the smallest nonempty \mathbf{G}_δ -set.

A set $F \subseteq X^\omega$ is dense in X^ω in case X^ω is the smallest closed set containing F , that is, $X^\omega \setminus F$ does not contain a nonempty open set. Since $\xi \in X^\omega$ is disjunctive, we have $\mathbf{T}(\xi) = X^*$, and therefore $\{\xi\} \cap X^* w X^\omega \neq \emptyset$ for all $w \in X^*$. Thus we have shown the following.

Proposition 9. An ω -word $\xi \in X^\omega$ is disjunctive if and only if the set $\{\xi\}$ is dense in X^ω in the topology of forbidden words.

This proposition shows that every closed set in the topology of forbidden words which contains some $\xi \in D$ must coincide with the whole space X^ω . Consequently, every \mathbf{F}_σ -set containing $\xi \in D$ equals X^ω .

Corollary 10. D is not an \mathbf{F}_σ -set in the topology of forbidden words.

Above we mentioned that the topology of forbidden words is not a metrizable topology, that is, it is not definable by a metric. Proposition 9 gives evidence of this fact, because the sets $\{\xi\}$, $\xi \in D$ are not closed, while in a metrizable topology every finite set must be closed.

5 A Metric Related to Languages

The definition of the topologies considered in this part is related to the well-known fact that every \mathbf{G}_δ -set of a complete metric space is a complete metric space itself (cf. [Ku66]), possibly using a different metric. We use here the construction presented in [St87]. Related investigations were carried out in [DNPY92].

As we have seen in Theorem 1, in Cantor space a \mathbf{G}_δ -set is of the form U^δ for some $U \subseteq X^*$. We use this language U to define a new metric ρ_U on X^ω which makes U^δ a closed set in the metric space (X^ω, ρ_U) :

$$\rho_U(\xi, \eta) = \begin{cases} 0 & , \text{ if } \xi = \eta, \text{ and} \\ r^{1-\#A(\xi) \cap A(\eta) \cap U} & , \text{ otherwise.} \end{cases} \quad (4)$$

This metric, in some sense, resembles the metric ρ in Cantor space; in fact, $\rho = \rho_{X^*}$. Moreover, since $\rho_U(\xi, \eta) \geq \rho(\xi, \eta)$, the U -topology refines the topology of the Cantor space. In particular, every closed set in Cantor space is also closed in the U -topology.

We denote by $C_U(F)$ the smallest closed (with respect to ρ_U) subset of X^ω containing F . A point $\xi \in C_U(F)$ is called an *isolated point of F* provided $\exists \varepsilon (\varepsilon > 0 \wedge \forall \eta (\eta \in F \wedge \eta \neq \xi \Rightarrow \rho_U(\xi, \eta) > \varepsilon))$. It should be mentioned that an arbitrary set of isolated points of X^ω is open.

A point $\xi \in C_U(F)$ which is not an isolated point of F is called an *accumulation point of F* .

Lemma 11 ([St03, Corollary 3]). *Let $U \subseteq X^*$. Then U^δ is the set of accumulation points of the whole space in (X^ω, ρ_U) .*

As an immediate consequence we obtain the following property of U^δ in the space (X^ω, ρ_U) which explains that the U -topology may be indeed finer than the topology of Cantor space.

Corollary 12. *If $F \supseteq U^\delta$ then F is a closed subset of (X^ω, ρ_U) .*

Proof. Lemma 11 shows that every point $\xi \in X^\omega \setminus F$ is an isolated point of X^ω . Consequently, $X^\omega \setminus F$ is open in (X^ω, ρ_U) . \square

It should be mentioned that, although U^δ is the set of accumulation points of the whole space (X^ω, ρ_U) , it may contain isolated points with respect to itself.

Example 4. *Let $U := a^* \cup a^*ba^* \subseteq \{a, b\}^*$. Then every ω -word $\xi \in a^*ba^\omega$ is an isolated point of $U^\delta = a^\omega \cup a^*ba^\omega$.*

In the case of the ω -language of disjunctive sequences, D , we can prove even more. To this end we mention the following relationship between accumulation points in U -topology and in Cantor space.

Lemma 13 ([St03, Theorem 4]). *Let $U \subseteq X^*$, $F \subseteq X^\omega$ and let $\xi \in U^\delta$. Then ξ is an accumulation point of F in (X^ω, ρ_U) if and only if ξ is an accumulation point of F in (X^ω, ρ) .*

In Proposition 3 we constructed a language W_D for which $D = W_D^\delta$. The following theorem shows that D is the set of its accumulation points, that is, in (X^ω, ρ_{W_D}) , D is closed and dense in itself.

Theorem 14. *Let $U^\delta = D$. In the space (X^ω, ρ_U) the ω -language D equals the set of its accumulation points.*

Proof. From Corollary 12 we know that D is closed in U -topology. Thus no point $\eta \notin D$ is an accumulation point of D .

On the other hand, since $w \in X^*$ and $\zeta \in D$ imply $w\zeta \in D$, every point $\xi \in D$ is an accumulation point of D in Cantor space. The assertion follows with Lemma 13. \square

This shows that in every space (X^ω, ρ_U) where $U^\delta = D$ the set of disjunctive sequences is the set of accumulation points of itself as well as the set of accumulation points of the whole space.

References

- [Ca02] Calude, C.S. *Information and Randomness: An Algorithmic Perspective*, 2nd Edition, Revised and Extended, Springer Verlag, Berlin, 2002.
- [CPS97] C. Calude, L. Priese and L. Staiger, *Disjunctive Sequences: An Overview*, CDMTCS Research Report 063, 1997.
- [DNPY92] Ph. Darondeau, D. Nolte, L. Priese and S. Yoccoz, Fairness, Distances and Degrees, *Theoret. Comput. Sci.* **97** (1992), 131–142.
- [En77] R. Engelking, *General Topology*. PWN – Polish Scientific Publishers, Warszawa 1977.
- [ET90] S.I. El-Zanati, W.R.R. Transue, On dynamics of certain Cantor sets, *J. Number Theory*, **36** (1990), 246–253.
- [He96] P. Hertling, Disjunctive ω -words and Real Numbers, *Journal of Universal Computer Science* **2** (1996) 7, 549 – 568.
- [JST83] H. Jürgensen, H.J. Shyr and G. Thierrin, Disjunctive ω -languages. *Elektron. Informationsverarb. Kybernetik* **EIK 19** (1983) 6, 267–278.
- [JT83] H. Jürgensen and G. Thierrin, On ω -languages whose syntactic monoid is trivial, *Intern. J. Comput. Inform Sci.* **12** (1983) 5, 359–365.
- [JT86] H. Jürgensen and G. Thierrin, Which monoids are syntactic monoids of ω -languages, *Elektron. Informationsverarb. Kybernetik* **EIK 22** (1986) 10/11, 513–536.
- [Ku66] K. Kuratowski, *Topology I*, Academic Press, New York, 1966.

- [MS97] O. Maler and L. Staiger, On syntactic congruences for ω -languages, *Theoret. Comput. Sci.* **183** (1997) 1, 93–112.
- [St83] L. Staiger, Finite-state ω -languages, *J. Comput. System. Sci.* **27** (1983), 434–448.
- [St87] L. Staiger, Sequential mappings of ω -languages. *RAIRO Infor. théor. et Appl.* **21** (1987) 2, 147–173.
- [St97] L. Staiger, ω -languages, in: *Handbook of Formal Languages* (G. Rozenberg and A. Salomaa Eds.), Vol. 3, Springer-Verlag, Berlin 1997. 339 – 387.
- [St02] L. Staiger, How large is the set of disjunctive sequences? *Journal of Universal Computer Science* **8** (2002) 2, 348–362.
- [St03] L. Staiger, Weighted Finite Automata and Metrics in Cantor Space, *J. Automata, Languages and Combinatorics*, **8** (2003) 2, 353 – 360.
- [Th90] W. Thomas, Automata on Infinite Objects, in: *Handbook of Theoretical Computer Science*, (J. Van Leeuwen Ed.), Vol. B, 133–191, Elsevier, Amsterdam, 1990.

Received October, 2001

On the Finiteness of Picture Languages of Synchronous Deterministic Chain Code Picture Systems

Bianca Truthe*

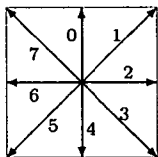
Abstract

Chain Code Picture Systems are LINDENMAYER systems over a special alphabet. The strings generated are interpreted as pictures. This leads to Chain Code Picture Languages. In this paper, synchronous deterministic Chain Code Picture Systems (*sDOL* systems) are studied with respect to the finiteness of their picture languages.

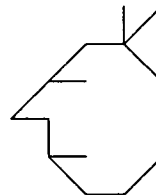
First, a hierarchy of abstractions is developed, in which the interpretation of a string as a picture passes through a multilevel process. Second, on the basis of this hierarchy, an algorithm is designed which decides the finiteness or infiniteness of any *sDOL* system in polynomial time.

1 Introduction

Important tasks in the area of picture processing are describing, creating, storing and recognizing pictures. With chain codes FREEMAN provided, in the 1960s, a possibility for describing line graphics [Fre74]. A picture is formed by a sequence of drawing commands that are represented by symbols (letters). A string describes a picture, which is built by the drawing commands of its letters. FREEMAN uses an alphabet $\{0, \dots, 7\}$, whose elements are interpreted according to the following sketch:



The picture to the right, for example, is generated by the word 1261204153445672606:
(For reconstructing start at the tip of the nose.)



This connection of strings and pictures suggests to search for relations between formal languages and picture sets. For language theoretical considerations the four directions $\{0, 2, 4, 6\}$ are sufficient, because the additional four do not yield

*Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, PSF 4120; D-39016 Magdeburg; Germany. E-mail: truthe@isg.cs.uni-magdeburg.de

completely different results nor require different methods to prove the decidability of finiteness [DH89].

According to plotter commands, r , u , l , d are written as the directions *right*, *up*, *left*, *down*. With chain codes, patterns like curves, fractals or folklore patterns can be described:

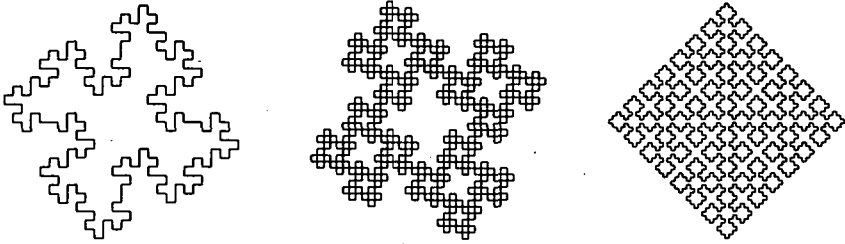


Figure 1: Applications of chain codes

Chain Code Picture Systems are LINDENMAYER systems over chain codes; in this connection, the picture languages generated are of interest.

This paper follows investigations on the decidability of the finiteness of picture languages generated by synchronous Chain Code Picture Systems (*sTOL* systems) presented by Dassow and Hromkovič in [DHR92]. That paper does not say anything about how many pictures are generated in the case of finiteness. During the work on this topic it turned out that synchronous deterministic Chain Code Picture Systems with the synchronization parameter $k = 1$ can generate finite or infinite picture languages, which is in contrast to a statement in [DHR92].

In this paper, conditions are obtained under which such a system generates a finite picture language or an infinite one. For this, a hierarchy of abstractions was developed such that the interpretation of a string as a picture passes through a multilevel process.

On this basis, a complete system of finiteness conditions is obtained such that one can decide, in polynomial time, for any *sDOL* system (with an arbitrary synchronization parameter), whether the picture language generated is finite or infinite, and how many pictures are generated in the case of finiteness.

2 Fundamentals

The finiteness investigations about picture languages of synchronous deterministic Chain Code Picture Systems are based on a hierarchy of abstractions. The lowest level covers the strings over the alphabet $\{r, l, u, d\}$. Graphs of different levels of abstraction, that represent various interpretations of the strings, are associated with the strings. Such a hierarchy exists for each Chain Code Picture System over the alphabet $\{r, l, u, d\}$. The lowest level contains the string set generated by the

system. The graph set of the highest level is regarded as the picture language generated by the system.

2.1 Structures over an Alphabet

Let $\mathcal{A} = \{r, l, u, d\}$ be an alphabet. The set \mathcal{A}^* is the set of all strings (with a finite length) over the alphabet \mathcal{A} that are built by concatenating letters of \mathcal{A} . The empty string is symbolized by λ ; the set \mathcal{A}^* without the empty string by \mathcal{A}^+ : $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\lambda\}$. The free structure (\mathcal{A}^*, \cdot) over the alphabet \mathcal{A} with the concatenation operation \cdot is a monoid.

The length $\#w$ of a string w is the number of letters in w . The set of all strings of length n from \mathcal{A}^* is denoted by \mathcal{A}^n . A string $w \in \mathcal{A}^n$ is composed of letters w_1, \dots, w_n if not stated otherwise: $w = w_1 \cdots w_n$. A substring $w_1 \cdots w_i$ ($0 \leq i \leq n$) is written as \vec{w}_i ($\vec{w}_0 = \lambda$).

For a string $w \in \mathcal{A}^n$ and a letter $x \in \mathcal{A}$, $\#_x w$ is the number of occurrences of x in w . For a string $w \in \mathcal{A}^*$, $[w]$ is the set of all letters in w :

$$[w] = \{x \mid \#_x w \geq 1\}.$$

The elements w of \mathcal{A}^* are interpreted as mappings on \mathbb{Z}^2 :

$$w : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2 \quad (w \in \mathcal{A}^*),$$

which are inductively defined as follows. The atomic mappings r, l, u, d assign to a point $q \in \mathbb{Z}^2$ its neighbours:

$$\begin{array}{ll} r(q) = q + (1, 0) & l(q) = q - (1, 0) \\ u(q) = q + (0, 1) & d(q) = q - (0, 1) \end{array}.$$

The translations $x(q) - q$ of any point $q \in \mathbb{Z}^2$ to its neighbours $x(q)$ are designated by $v_x \in \mathbb{Z}^2$:

$$v_x = \begin{cases} (1, 0), & \text{if } x = r \\ (-1, 0), & \text{if } x = l \\ (0, 1), & \text{if } x = u \\ (0, -1), & \text{if } x = d \end{cases}.$$

The mappings x from \mathcal{A} are translations. Every mapping $x \in \mathcal{A}$ is surjective (the range of values is \mathbb{Z}^2), injective (from $x(p) = x(q)$ always follows $p = q$) and, therefore, bijective (one to one).

Two arbitrary mappings x, y are called disjoint if their function values differ for each argument.

Proposition 1. *Every two different mappings $x, y \in \mathcal{A}$, ($x \neq y$), never give the same neighbour: $\forall q \in \mathbb{Z}^2 : x(q) \neq y(q)$. This means that the mappings in \mathcal{A} are disjoint.*

The empty string corresponds to the identical mapping

$$\lambda : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2 \quad \text{with } q \mapsto q.$$

A compound string $w \in \mathcal{A}^*$ stands for the concatenated mapping $v \circ w$:

$$v \circ w : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2 \quad \text{with } q \mapsto w(v(q)).$$

The zero point of the \mathbb{Z}^2 is symbolized by o : $o = (0, 0)$.

This interpretation of strings as mappings on \mathbb{Z}^2 is a homomorphism from the free structure (\mathcal{A}^*, \cdot) in the free structure (\mathcal{A}^*, \circ) . Hence (\mathcal{A}^*, \circ) is also a monoid. For each mapping $w \in \mathcal{A}^*$, an inverse mapping $w^{-1} \in \mathcal{A}^*$ exists:

- The inverse of the identical mapping is the identical mapping: $\lambda^{-1} = \lambda$.
- The inverses of the atomic mappings are $r^{-1} = l$, $l^{-1} = r$, $u^{-1} = d$, $d^{-1} = u$, because $v_x = -v_{x-1}$ ($x \in \mathcal{A}$).
- Let $w = w_1 \circ \dots \circ w_n$ ($w_i \in \mathcal{A}$, $i = 1, \dots, n$) be a concatenated mapping. Then the inverse mapping is $w^{-1} = w_n^{-1} \circ \dots \circ w_1^{-1}$.

This result is stated in the following proposition.

Proposition 2. *The algebraic structure (\mathcal{A}^*, \circ) is a group.*

The operator \circ is not written if the context shows which operation is meant. For example, $(x_1 x_2)(o)$ implies that $x_1 x_2$ symbolizes the concatenated mapping $x_1 \circ x_2$, whereas $x_1 x_2$ in $[x_1 x_2]$ represents the compound string $x_1 x_2$.

The mappings ru , ur and ld , dl assign the diagonal neighbours to a point q :

$$ru(q) = ur(q) = q + (1, 1); \quad ld(q) = dl(q) = q - (1, 1).$$

These relations are symbolized by $-$ and $^\perp$:

x	\bar{x}	x^\perp	\bar{x}^\perp
r	l	u	d
l	r	d	u
u	d	r	l
d	u	l	r

The mappings $w \in \mathcal{A}^n$ are translations: $w(p + q) = w(p) + q$ (can be proved by induction over n). This leads to the following proposition about the correlation between mappings of the zero point.

Proposition 3. *Let v , w be two words of \mathcal{A}^* . The mapping of the zero point by the compound mapping $v \circ w$ is $(v \circ w)(o) = v(o) + w(o)$.*

Proof. The mapping of the zero point by $v \circ w$ yields



$$\begin{aligned} (v \circ w)(o) &= w(v(o)) \\ &= w(o + v(o)) \\ &= w(o) + v(o) \\ &= v(o) + w(o). \end{aligned}$$

□

2.2 Graphical Embedding

A grid graph is a graph with the following properties:

- The set of vertices is a subset of \mathbb{Z}^2 .
- Each edge connects two neighbours $q \in \mathbb{Z}^2$ and $x(q)$ with $x \in \mathcal{A}$.

The position of the vertices is essential; renaming of the vertices does not yield an isomorphic graph. For example, the graphs  and  should be considered as non-identical.

For each point $a \in \mathbb{Z}^2$, functions exist that assign, to a word $w \in \mathcal{A}^n$

- the set of vertices $\odot^a(w) = \{ \vec{w}_i(a) \mid i = 0, \dots, n \}$,
- the directed grid graph (possibly with multiple edges)

$$g^a(w) = \left(\odot^a(w), \{ (\vec{w}_{i-1}(a), \vec{w}_i(a)) \}_{i=1, \dots, n} \right),$$

- the simple directed grid graph $s^a(w)$ of $g^a(w)$ (without multiple edges),
- the set of edges $\|^a w$ of $s^a(w)$ in a different notation

$$\|^a w = \{ (\vec{w}_{i-1}(a), w_i) \mid i = 1, \dots, n \},$$

- the picture (the shade of $s^a(w)$)

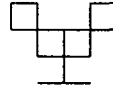
$$p^a(w) = (\odot^a(w), \{ (\vec{w}_{i-1}(a), \vec{w}_i(a)), (\vec{w}_i(a), \vec{w}_{i-1}(a)) \mid i = 0, \dots, n \}).$$

If the reference point a is the zero point, the upper index will be omitted. The set $\|^a w$ contains a pair (q, x) with $q \in \mathbb{Z}^2$ and $x \in \mathcal{A}$ if and only if $(q, x(q))$ is an edge in the graph $s^a(w)$ (if (q, \tilde{q}) is an edge then $x \in \mathcal{A}$ with $x(q) = \tilde{q}$ exists uniquely - due to Prop. 1). Thus, the graph $s^a(w)$ is one to one associated with the set $\|^a w$. Throughout this paper, this set is referred to as the edge set of w with respect to a .

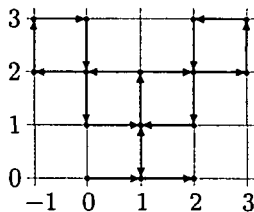
The following example shall demonstrate these correlations:

Example 1. Let $w = ruullurddrurrulddldr$ be a word from \mathcal{A}^* .

If a plotter takes this word as a sequence of elementary commands for drawing, the resulting picture will be:



For investigations on picture languages generated by *sDOL* systems, it is necessary also to know *how* a picture is drawn. To show how a pictured arises, the lines are marked by arrows indicating the drawing direction. Additionally, the grid points are marked (the only points where the direction can change). This leads to the following grid graph (beginning at the zero point). Note that the line from $(1,1)$ upwards is drawn twice.



The vertex set $\odot(w)$ contains all grid points visited:

$o, r(o) = (1, 0), ru(o) = (1, 1), ruu(o) = (1, 2), \dots, w(o) = (2, 0)$.

The directed grid graph $g^a(w)$ consists of the vertex set $\odot(w)$ and all edges on the ‘drawing path’: $(o, (1, 0)), ((1, 0), (1, 1)), ((1, 1), (1, 2)), \dots, ((0, 1), (1, 1)), ((1, 1), (1, 2)), ((1, 2), (2, 2)), \dots, ((1, 0), (2, 0))$. The edge $((1, 1), (1, 2))$ occurs twice because the underlined letters in $ru\underline{u}llurddr\underline{u}rrulddldr$ both produce this line (because of $ru(o) = (1, 1) = ru\underline{u}llurddr(o)$). The edge set $\|w$ consists of all edges passed as elements of $\mathbb{Z}^2 \times \mathcal{A}$ instead of $\mathbb{Z}^2 \times \mathbb{Z}^2$: $(o, r), ((1, 0), u), ((1, 1), u), \dots, ((1, 1), d), ((1, 0), r)$.

Since the pictures are shades of the simple directed graphs, one immediately notices that two words having the same edge set also represent the same picture. This result is stated in the following proposition, so it can be referred to.

Proposition 4. *If the edge sets $\|v$ and $\|w$ of two words $v, w \in \mathcal{A}^*$ coincide, so the pictures $p(v)$ and $p(w)$ do as well.*

The following proposition states correlations between concatenating strings and combining graphs.

Proposition 5. *The concatenation of strings is associated with a union of vertex sets, directed graphs, edge sets, and pictures: For each point $a \in \mathbb{Z}^2$ and any two strings $v, w \in \mathcal{A}^*$, one has*

$$\begin{aligned}\odot^a(vw) &= \odot^a(v) \cup \odot^{v(a)}(w), \\ g^a(vw) &= g^a(v) \cup g^{v(a)}(w), \\ \|^{a}vw &= \|^{a}v \cup \|^{v(a)}w, \\ p^a(vw) &= p^a(v) \cup p^{v(a)}(w).\end{aligned}$$

Proof. Let v be an element of \mathcal{A}^n and w be an element of \mathcal{A}^m . Then the union of the vertex sets $\odot^a(v)$ and $\odot^{v(a)}(w)$ is

$$\begin{aligned}\odot^a(v) \cup \odot^{v(a)}(w) &= \{ \vec{v}_i(a) \mid i = 0, \dots, n \} \cup \{ \vec{w}_i(v(a)) \mid i = 0, \dots, m \} \\ &= \{ a, v_1(a), \dots, v(a), w_1(v(a)), \dots, vw(a) \} \\ &= \odot^a(vw).\end{aligned}$$

The union of the other sets can be shown similarly. (Note that the non-simple graphs possibly contain multiple edges.) \square

The sets

- \mathcal{A}^* of strings,
- $\mathcal{G} = \{ g^a(w) \mid w \in \mathcal{A}^*, a \in \mathbb{Z}^2 \}$ of directed graphs,
- $\mathcal{S} = \{ s^a(w) \mid w \in \mathcal{A}^*, a \in \mathbb{Z}^2 \}$ of simple directed graphs, and
- $\mathcal{P} = \{ p^a(w) \mid w \in \mathcal{A}^*, a \in \mathbb{Z}^2 \}$ of pictures

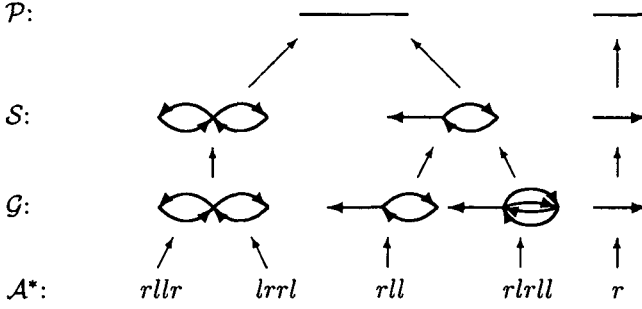


Figure 2: Hierarchy of abstractions

form a hierarchy of different levels of abstraction. A part of this hierarchy is to be seen in Figure 2.

A later derivation of words by a simultaneous replacing of letters can be interpreted as a derivation of graphs by replacing an edge by a graph. In deterministic systems, one letter is always replaced by the same word. Thus, one edge is always replaced by the same graph – a derivation of an edge is independent from the number of its occurrences. In non-deterministic systems however, a letter at one position can be replaced by a different word than the same letter at another position. Hence, one occurrence of an edge can be replaced by a different graph than another occurrence of the same edge – the number of occurrences is essential. For this reason, the graphs with multiple edges are kept in the hierarchy, although the simple graphs are sufficient in this paper.

A rectangle R determined by two points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ is the set of all points $a = (a_x, a_y)$ between p and q :

$$R = \left\{ a \in \mathbb{Z}^2 \mid \begin{array}{l} p_x \leq a_x \leq q_x \text{ or } q_x \leq a_x \leq p_x \text{ and} \\ p_y \leq a_y \leq q_y \text{ or } q_y \leq a_y \leq p_y \end{array} \right\}.$$

Such a rectangle is written as $R = [p, q]$. The picture area of a set $S \subseteq \mathbb{Z}^2$, denoted by \mathfrak{R}_S , is the smallest rectangle that contains S . By scaling a picture area $\mathfrak{P} = [p, q]$ by a factor $s \in \mathbb{N}_0$, the picture area $s\mathfrak{P} = \{sb \mid b \in \mathfrak{P}\} = [sp, sq]$ is obtained. The union of two picture areas is not a rectangle in general. The extended union shall give the rectangle covering the normal union:

$$\mathfrak{R}_P \cup \mathfrak{R}_Q = \mathfrak{R}_{P \cup Q}.$$

Let $\odot^a(w)$ be the vertex set of a word w with respect to a . Then, the functions $\lfloor \cdot \rfloor_w^a, \lceil \cdot \rceil_w^a, \lfloor \cdot \rfloor_w^a, \lceil \cdot \rceil_w^a$ give the 'border' of $\odot^a(w)$:

$$\begin{aligned} \lfloor \cdot \rfloor_w^a &= \min \{ x \mid (x, y) \in \odot^a(w) \}, & \lceil \cdot \rceil_w^a &= \min \{ y \mid (x, y) \in \odot^a(w) \} \\ \lfloor \cdot \rfloor_w^a &= \max \{ x \mid (x, y) \in \odot^a(w) \}, & \lceil \cdot \rceil_w^a &= \max \{ y \mid (x, y) \in \odot^a(w) \} \end{aligned}$$

The symbols $\lfloor \cdot \rfloor_w^a = (\lfloor \cdot \rfloor_w^a, \lceil \cdot \rceil_w^a)$ and $\lceil \cdot \rceil_w^a = (\lfloor \cdot \rfloor_w^a, \lceil \cdot \rceil_w^a)$ stand for the lower-left and upper-right corners. The picture area of $\odot^a(w)$ is denoted by $\square^a(w) = [\lfloor \cdot \rfloor_w^a, \lceil \cdot \rceil_w^a]$.

2.3 Special Endomorphisms

Let κ, μ be two natural numbers, $\kappa, \mu \in \mathbb{N}_0$. An endomorphism h on \mathcal{A}^* is called (κ, μ) -endomorphism if the following conditions are satisfied for each $x \in \mathcal{A}$:

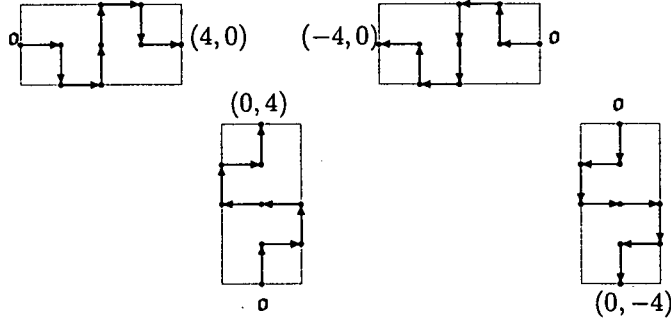
1. $(h(x))(o) = \kappa v_x$.
2. $\square(h(x)) \subseteq \kappa[o, v_x] \uplus \mu[v_{x^\perp}, v_{\bar{x}^\perp}]$.

The following example illustrates this.

Example 2. Let h be an endomorphism on \mathcal{A}^* with $r \mapsto rdruurdr$, $l \mapsto lulddlul$, $u \mapsto urulluru$, and $d \mapsto dldrrdld$. Then one has

$$\begin{aligned} rdruurdr(o) &= 4v_r + 2v_d + 2v_u = 4v_r, \\ lulddlul(o) &= 4v_l + 2v_u + 2v_d = 4v_l, \\ urulluru(o) &= 4v_u + 2v_r + 2v_l = 4v_u, \\ dldrrdld(o) &= 4v_d + 2v_l + 2v_r = 4v_d. \end{aligned}$$

Hence, the first condition is satisfied. The simple directed graphs of $h(x)$ ($x \in \mathcal{A}$) are:



All points of $\odot(h(x))$ (for each $x \in \mathcal{A}$) lie in the rectangle covering both the lines $[o, 4v_x]$ and $[v_{x^\perp}, v_{\bar{x}^\perp}]$. The picture area is $\square(h(x)) = 4[o, v_x] \uplus [v_{x^\perp}, v_{\bar{x}^\perp}]$ for $x \in \mathcal{A}$. Thus, h is a $(4, 1)$ -endomorphism. The first synchronization condition says where the end point of a drawing lies. The second one causes the pictures to lie in certain rectangles.

The n -ary concatenation of an endomorphism h is written shortly as h^n . Applied to a string $w \in \mathcal{A}^*$, its result is the n -th derivative of w ; written as $w^{(n)}$ (w' , w'' , w''' for the first three derivatives, $w^{(0)} = w$). The parameter κ defines the length of the derivation picture regarding the respective direction; the derivation is length-contracting in the case of $\kappa < 1$, length-constant in the case of $\kappa = 1$, and length-expanding in the case of $\kappa > 1$. The parameter μ is the width of the derivation picture.

2.4 Chain Code Picture Systems

A synchronous deterministic context free Chain Code Picture System (*sDOL* system) is a triple

$$G = (\mathcal{A}, h, \omega)$$

with the alphabet $\mathcal{A} = \{r, l, u, d\}$, a (κ, μ) -endomorphism h on \mathcal{A}^* , and a non-empty start string (axiom) $\omega \in \mathcal{A}^+$.

The picture language P_G generated by an *sDOL* system G is the set of all pictures of derivatives of the axiom ω :

$$P_G = \left\{ p(\omega^{(n)}) \mid n \in \mathbb{N}_0 \right\}.$$

An *sDOL* system is called length-contracting (-constant, -expanding) if the (κ, μ) -endomorphism belonging to it, has this property.

3 Finiteness Investigations

Let $G = (\mathcal{A}, h, \omega)$ be an *sDOL* system with a (κ, μ) -endomorphism h . The n -th derivative ($n \in \mathbb{N}$) of any letter $x \in \mathcal{A}$ maps the zero point \mathbf{o} to the point $\kappa^n \mathbf{v}_x$: $x^{(n)}(\mathbf{o}) = \kappa^n \mathbf{v}_x$. This can be proved by induction.

Moreover, the first synchronization condition says that $x'(\mathbf{o}) = \kappa \mathbf{v}_x$. This means $x'(\mathbf{o}) = \kappa \mathbf{v}_x + c \mathbf{v}_x + c \mathbf{v}_{\bar{x}} + d \mathbf{v}_{x^\perp} + d \mathbf{v}_{\bar{x}^\perp}$ for some natural numbers c, d . Consequently, x^\perp and \bar{x}^\perp have the same numbers of occurrences in the derivative x' , and x has κ more occurrences than \bar{x} . These observations are summarized in the following proposition.

Proposition 6. *For all $x \in \mathcal{A}$, one has*

1. $x^{(n)}(\mathbf{o}) = \kappa^n \mathbf{v}_x$ for $n \in \mathbb{N}$,
2. $\#_x x' = \kappa + \#_{\bar{x}} x'$,
3. $\#_{x^\perp} x' = \#_{\bar{x}^\perp} x'$.

The Chain Code Picture Systems are distinguished by their ‘length behaviour’ (represented by the parameter κ).

3.1 Length-contracting Chain Code Picture Systems

Let $G = (\mathcal{A}, h, \omega)$ be an *sDOL* system with a length-contracting (κ, μ) -endomorphism h . Since $\kappa < 1$ and $\kappa \in \mathbb{N}_0$, κ must be equal to 0.

The second synchronization condition has the effect that the picture area of the derivative x' of a letter $x \in \mathcal{A}$ is a line: $\square(x') \subseteq \mu[\mathbf{v}_x^\perp, \mathbf{v}_{\bar{x}^\perp}]$. Hence, x and \bar{x} do not occur in the string x' .

For any string $w \in \mathcal{A}^*$, exactly one of the following three cases occurs:

1. $w' = \lambda$. Then all further derivatives are also empty: $w^{(n)} = \lambda$, $n \geq 1$.

2. $w' \neq \lambda$, $w'' = \lambda$. Then all further derivatives are empty: $w^{(n)} = \lambda$, $n \geq 2$.
3. $w' \neq \lambda$, $w'' \neq \lambda$. Since $w' \neq \lambda$, some letter $x \in \mathcal{A}$ occurs in w' and also the same number of \bar{x} . Hence, the letter set $[w']$ can be $\{r, l\}$, $\{u, d\}$ or \mathcal{A} .

Example 3. For example, let h be a $(0, \mu)$ -endomorphism with

$$r \mapsto ud, u \mapsto rl, d \mapsto \lambda \text{ and } l \mapsto \lambda.$$

If $w = r$ then $[w'] = \{u, d\}$, if $w = u$ then $[w'] = \{r, l\}$. If $w = ru$ then the letter set $[w']$ consists of all letters: $[w'] = \mathcal{A}$.

If $[w'] = \{r, l\}$, the letter set $[w'']$ is $\{u, d\}$ (it cannot be empty because $w'' \neq \lambda$). Similarly, $[w''] = \{r, l\}$ if $[w'] = \{u, d\}$. If $[w'] = \mathcal{A}$, some letter $x \in \mathcal{A}$ occurs in the word w together with x^\perp or \bar{x}^\perp . As these letters occur in w' also, w'' consists of the same letters as w' . Summerized, one has

$$[w''] = \begin{cases} \{u, d\} & \text{if } [w'] = \{r, l\}, \\ \{r, l\} & \text{if } [w'] = \{u, d\}, \\ \mathcal{A} & \text{if } [w'] = \mathcal{A}. \end{cases}$$

An analogous argumentation leads to

$$[w'''] = \begin{cases} \{r, l\} & \text{if } [w'] = \{r, l\}, \\ \{u, d\} & \text{if } [w'] = \{u, d\}, \\ \mathcal{A} & \text{if } [w'] = \mathcal{A}, \end{cases}$$

that is $[w'''] = [w']$. Thus, the letter set of the fourth derivative coincides with that one of the second derivative:

$$[w^{(4)}] = \bigcup_{x \in [w''']} [x'] = \bigcup_{x \in [w'']} [x'] = [w''].$$

From this case distinction one can conclude the following proposition by induction:

Proposition 7. *The letter sets from the second derivative of a string $w \in \mathcal{A}^*$ on are either empty or alternate beginning with the first derivative:*

$$[w''] = \emptyset \implies [w^{(n)}] = \emptyset \quad (n \geq 2),$$

$$[w''] \neq \emptyset \implies [w^{(2n-1)}] = [w'] \wedge [w^{(2n)}] = [w''] \quad (n \geq 2).$$

A similar correlation can be found for edge sets. Let $w^{(n)}$ be the n -th derivative of a string $w \in \mathcal{A}^*$:

$$w^{(n)} = x_1 \cdots x_l \quad (x_i \in \mathcal{A}, i = 1, \dots, l).$$

The edge set of $w^{(n+1)}$ is, according to Proposition 5,

$$\|w^{(n+1)}\| = \|x'_1 \cup \|x'_1{}^{(o)}x'_2 \cup \dots \cup \|x'_1 \cdots x'_{l-1}{}^{(o)}x'_l\|.$$

The Propositions 3 and 6 imply that $(x'_1 \cdots x'_i)(o) = x'_1(o) + \cdots + x'_i(o) = o$ for $i = 1, \dots, l$. Hence, the edge set of $w^{(n+1)}$ is

$$\|w^{(n+1)} = \bigcup_{i=1}^l \|x'_i = \bigcup_{x \in \|w^{(n)}} \|x' \quad (\text{if } \hat{x} = \hat{x} \text{ then } \hat{x}' = \hat{x}' \text{ and } \|\hat{x}' = \|\hat{x}').$$

According to Proposition 7, one obtains $\|w^{(n)} = \emptyset$ if $w'' = \lambda$ and, in the case that $w'' \neq \lambda$:

$$\begin{aligned} \|w^{(2n)} &= \bigcup_{x \in \|w^{(2n-1)}} \|x' = \bigcup_{x \in \|w'} \|x' = \|w'', \\ \|w^{(2n+1)} &= \bigcup_{x \in \|w^{(2n)}} \|x' = \bigcup_{x \in \|w''} \|x' = \|w''', \end{aligned}$$

for $n \geq 2$.

Thus, if $w'' = \lambda$ then the pictures of $w^{(n)}$ ($n \geq 2$) consist of the zero point only. If $w'' \neq \lambda$ then the pictures of even derivations coincide from the second derivation on; those of odd derivations from the third one on (because of Prop. 4). After the third derivation of a word $w \in \mathcal{A}^*$, no new picture arises.

Theorem 1. *Let $G = (\mathcal{A}, h, \omega)$ be a length-contracting sDOL system. The picture language generated is*

$$P_G = \{ p(\omega), p(\omega'), p(\omega''), p(\omega''') \}.$$

The following example shows a length-contracting sDOL system together with its picture language.

Example 4. Let $G = (\mathcal{A}, h, r)$ be an sDOL system with a $(0, \mu)$ -endomorphism h with $r \mapsto ud$, $l \mapsto du$, $u \mapsto rl$, and $d \mapsto \lambda$. The words generated by G are r , ud , rl , $uddu$, $(rl)^2$, $(uddu)^2$, $(rl)^4$, $(uddu)^4$, etc. The corresponding pictures are

$\longrightarrow, \begin{array}{|c|} \hline l \\ \hline \end{array}, \longrightarrow, \begin{array}{|c|} \hline l \\ \hline \end{array}, \longrightarrow, \begin{array}{|c|} \hline l \\ \hline \end{array}, \longrightarrow, \begin{array}{|c|} \hline l \\ \hline \end{array}, \text{ etc.}$

Since a new picture does not occur, the picture language is $\left\{ \longrightarrow, \begin{array}{|c|} \hline l \\ \hline \end{array} \right\}$.

3.2 Length-expanding Chain Code Picture Systems

Let $G = (\mathcal{A}, h, \omega)$ be a length-expanding sDOL system. Since the (κ, μ) -endomorphism h is length-expanding, κ is greater than 1.

Let $x \in \mathcal{A}$ be the initial letter of ω (since $\omega \in \mathcal{A}^+$ it has at least one letter): $\omega = xw$ ($w \in \mathcal{A}^*$). The n -th derivative is $\omega^{(n)} = x^{(n)}w^{(n)}$, and $x^{(n)}(o)$ is a vertex of every graph of $\omega^{(n)}$:

$$x^{(n)}(o) \in \odot(\omega^{(n)}), \quad n \in \mathbb{N}_0.$$

Furthermore, let X_ω be the set of all points $x^{(n)}(o)$, V_ω the union of the vertex sets $\odot(\omega^{(n)})$, and P_ω the set of all pictures $p(\omega^{(n)})$, $n \in \mathbb{N}_0$:

$$X_\omega = \left\{ x^{(n)}(o) \mid n \in \mathbb{N}_0 \right\} = \left\{ k^n v_x \mid n \in \mathbb{N}_0 \right\} \quad (\text{Prop. 6}),$$

$$V_\omega = \bigcup_{n \in \mathbb{N}_0} \odot(\omega^{(n)}),$$

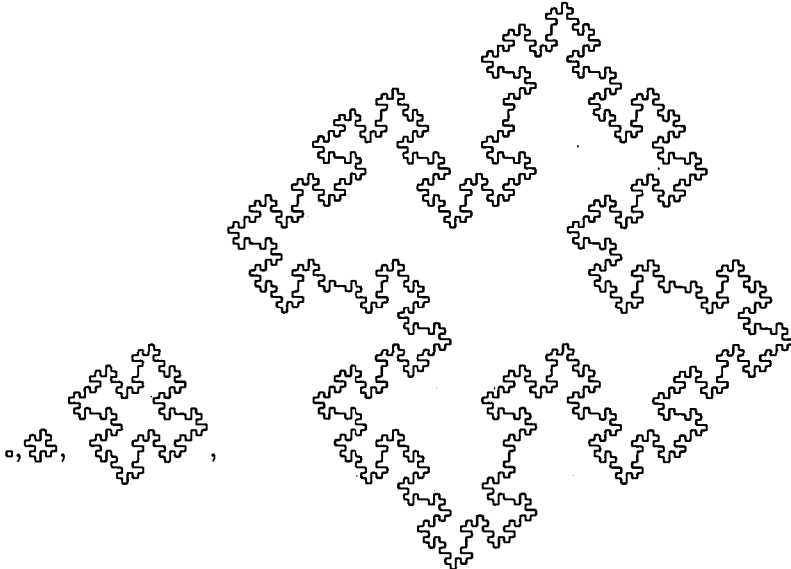
$$P_\omega = \left\{ p(\omega^{(n)}) \mid n \in \mathbb{N}_0 \right\}.$$

The set X_ω is infinite because $\kappa > 1$ and $v_x \neq o$. Every point of X_ω also occurs in V_ω : $X_\omega \subseteq V_\omega$; thus the set V_ω is also infinite. Each vertex set $\odot(\omega^{(n)})$, $n \in \mathbb{N}_0$, is finite; hence, there are infinitely many different ones in the union V_ω . If the vertex sets of two strings $u, v \in \mathcal{A}^*$ differ, so the pictures do as well. Hence, among the pictures $p(\omega^{(n)})$ with $n \in \mathbb{N}_0$, there are infinitely many different ones: $|P_\omega| = \infty$.

Theorem 2. *For every length-expanding sDOL system $G = (\mathcal{A}, h, \omega)$, the picture language P_G generated by G is infinite.*

The following example shows a length-expanding sDOL system.

Example 5. Let $G = (\mathcal{A}, h, \text{rul}d)$ be a length-expanding sDOL system with the $(4, 1)$ -endomorphism h given in Example 2. The first pictures generated are drawn below (up to the third derivation).



3.3 Length-constant Chain Code Picture Languages

In contrast to the previous situations, among the length-constant *sDOL* systems, there are those with a finite picture language as well as those with an infinite one. The following example shows an *sDOL* system with a finite picture language and a similar *sDOL* system with an infinite picture language.

Example 6. Let h be a $(1, \mu)$ -endomorphism with $r \mapsto rud$, $u \mapsto uldru$, $d \mapsto d$, and $l \mapsto l$. Then $G = (\mathcal{A}, h, r)$ is a length-constant *sDOL* system. The simple directed graph of the axiom is \rightarrow , of the first derivative $\rightarrow \uparrow$, and of the second derivative $\rightarrow \uparrow \rightarrow$. The graphs of the later derivatives are the same: $\rightarrow \uparrow \rightarrow$. Hence, the picture language generated is finite: $P_G = \{ \rightarrow, \rightarrow \uparrow, \rightarrow \uparrow \rightarrow \}$. Now, change h such that $l \mapsto lrl$. Then, the simple directed graph of the third derivative is not the same as that one of the second derivative, but $\rightarrow \uparrow \rightarrow$. A new r -edge arose that will be replaced (in the next step) by its derivative; thus, the fourth derivative has the graph $\rightarrow \uparrow \rightarrow \uparrow$. A new r -edge occurs in every third derivative (6th, 9th, etc.). Hence, the picture sizes increase; the picture language is infinite.

The example above shows that, in the case of $\kappa = 1$, further investigations are needed to find out when an *sDOL* system generates a finite picture language and when it does not.

Several examples lead to the supposition that the difference between the edge sets of the second and third derivatives indicates the finiteness of the picture language: If they do not differ, the picture language is finite; if there is a difference, then the language is infinite. This supposition will be confirmed and proved.

Let $G = (\mathcal{A}, h, \omega)$ be a length-constant *sDOL* system. The next proposition extends the first statement of Proposition 6 to words.

Proposition 8. *If h is a $(1, \mu)$ -endomorphism, then $w^{(n)}(\mathbf{o}) = w(\mathbf{o})$ for every string $w \in \mathcal{A}^*$ and every derivation step $n \in \mathbb{N}_0$.*

Proof. Let $w \in \mathcal{A}^l$ be a word $w_1 \cdots w_l$. Then, one can conclude the following equations:

$$\begin{aligned} w^{(n)}(\mathbf{o}) &= (w_1^{(n)} \cdots w_l^{(n)})(\mathbf{o}) \\ &= w_1^{(n)}(\mathbf{o}) \cdots w_l^{(n)}(\mathbf{o}) && \text{(Prop. 3)} \\ &= v_{w_1} + \cdots + v_{w_l} && \text{(Prop. 6)} \\ &= w_1(\mathbf{o}) + \cdots + w_l(\mathbf{o}) \\ &= w(\mathbf{o}). \end{aligned}$$

This proves the proposition. □

Let $w \in \mathcal{A}^*$ be a word. The operator $\|\cdot\|$ gives the edge set $\|w\|$ of w . Applying the $(1, \mu)$ -endomorphism h for n times produces the n -th derivative $w^{(n)}$; its edge set is $\|w^{(n)}\|$. The following proposition shows how the edge set $\|w^{(n)}\|$ can be obtained from the edge set $\|w\|$.

$$\begin{array}{ccc} w & \xrightarrow{\|\cdot\|} & \|w\| \\ \downarrow h & & \downarrow ? \\ w^{(n)} & \xrightarrow{\|\cdot\|} & \|w^{(n)}\| \end{array}$$

Proposition 9. *The edge set $\|w^{(n)}\|$ of the n -th derivative of a string $w \in \mathcal{A}^*$ is the union of the n -th derivatives of all edges in $\|w\|$:*

$$\|w^{(n)}\| = \bigcup_{(q,x) \in \|w\|} \|q x^{(n)}\| \quad w \in \mathcal{A}^*, n \in \mathbb{N}_0.$$

Proof. Let $w \in \mathcal{A}^l$ be the string $w_1 \cdots w_l$. Then the n -th derivative of w is

$$w^{(n)} = w_1^{(n)} \cdots w_l^{(n)}.$$

The edge set of w is

$$\begin{aligned} \|w\| &= \|w_1\| \cup \|\overrightarrow{w_1}(o)\| w_2 \cup \dots \cup \|\overrightarrow{w_{l-1}}(o)\| w_l \quad (\text{Prop. 5}) \\ &= \{ (o, w_1) \} \cup \{ (\overrightarrow{w_1}(o), w_2) \} \cup \dots \cup \{ (\overrightarrow{w_{l-1}}(o), w_l) \} \\ &= \{ (o, w_1), (\overrightarrow{w_1}(o), w_2), \dots, (\overrightarrow{w_{l-1}}(o), w_l) \}, \end{aligned}$$

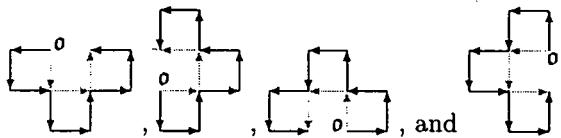
the edge set of $w^{(n)}$ is

$$\begin{aligned} \|w^{(n)}\| &= \|w_1^{(n)}\| \cup \|\overrightarrow{w_1^{(n)}}(o)\| w_2^{(n)} \cup \dots \cup \|\overrightarrow{w_{l-1}^{(n)}}(o)\| w_l^{(n)} \quad (\text{Prop. 5}) \\ &= \|w_1^{(n)}\| \cup \|\overrightarrow{w_1}(o)\| w_2^{(n)} \cup \dots \cup \|\overrightarrow{w_{l-1}}(o)\| w_l^{(n)} \quad (\text{Prop. 8}) \\ &= \bigcup_{(q,x) \in \|w\|} \|q x^{(n)}\|, \end{aligned}$$

which proves the proposition. \square

The simple directed graph of the n -th derivative of a word $w \in \mathcal{A}^*$ arises from the simple directed graph of w by replacing each x -edge $(q, x) \in \|w\|$ by the simple directed graph of the n -th derivative of x , beginning at the point q .

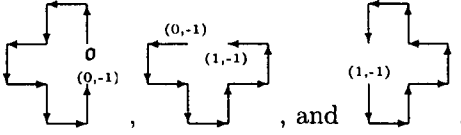
Example 7. Let h be a $(1, \mu)$ -endomorphism with $r \mapsto dru$, $u \mapsto rul$, $l \mapsto uld$, and $d \mapsto ldr$. The simple directed graphs of the second atomic derivatives are shown below (those of the first derivatives are inserted grey coloured):



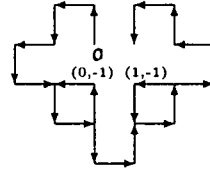
Let w be the first derivative of r . Its edge set $\|dru$ is

$$\{(\sigma, d), ((0, -1), r), ((1, -1), u)\}.$$

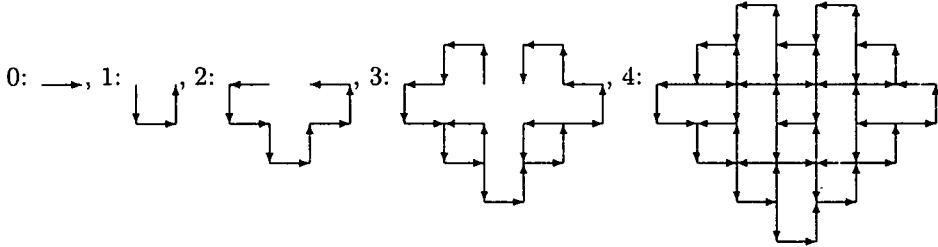
The edge set of the second derivative of w consists of all edges of the edge sets of the second derivatives of d with respect to σ , of r with respect to $(0, -1)$, and of u with respect to $(1, -1)$: $\|w'' = \|\sigma d'' \cup \|(0, -1)r'' \cup \|(1, -1)u''$. These edge sets are (shown as simple directed graphs):



Hence, the the simple directed graph of w'' is



Since w is the first derivative of r , this graph should be the same as that one of the third derivative of r – and is as the following sequence of derivative graphs shows.



With the help of the proposition above, the following proposition about the stability of edge sets can be proved.

Proposition 10. *If the edge set of a string $w \in \mathcal{A}^*$ coincides with that of its derivative w' , then it coincides with the edge set of every higher derivative:*

$$\|w = \|w' \implies \|w = \|w^{(n)}, \quad w \in \mathcal{A}^*, n \in \mathbb{N}.$$

Proof. The proof is carried out by induction. Suppose, that $\|w = \|w^{(i)}$ for $1 \leq i \leq n$. Then the edge set $\|w^{(n+1)}$ is

$$\begin{aligned} \|w^{(n+1)} &= \bigcup_{(q, x) \in \|w^{(n)}} \|q x' && \text{(Prop. 9)} \\ &= \bigcup_{(q, x) \in \|w} \|q x' && \text{(inductional assumption)} \\ &= \|w' && \text{(Prop. 9)} \\ &= \|w && \text{(inductional assumption)} \end{aligned}$$

from which the proposition follows. \square

With the Propositions 10 and 4, the first supposition (see page 65) is proved. So, it is stated in a lemma.

Lemma 1. *Let $G = (\mathcal{A}, h, \omega)$ be a length-constant sDOL system. If the edge sets of the second and third derivatives of the axiom ω coincide, then the picture language generated consists of the pictures up to the second derivative at most. That is shortly written as*

$$\|\omega'' = \|\omega''' \implies P_G = \{p(\omega), p(\omega'), p(\omega'')\}.$$

In the first example of this section (p. 65), one can observe that if the edge sets of the second and third derivatives of the axiom ω do not coincide, then at least one x -edge exists in the second derivative which is later replaced by a graph that contains another x -edge. The next proposition gives an even stronger restriction.

Proposition 11. *If the edge sets of the second and third derivatives of the axiom ω do not coincide, then there exists a letter $x \in [\omega'']$ such that one of the first three derivation edge sets $\|x'$, $\|x''$ or $\|x'''$ contains an edge different from (o, x) .*

Proof. The statement of the proposition is equivalent to the following statement. If, for each letter $x \in [\omega'']$, the edge sets of the first three derivatives of x do not contain any x -edge different from (o, x) then the edge sets of ω'' and ω''' coincide:

$$(\forall x \in [\omega''] : \|_x x = \|_x x' = \|_x x'' = \|_x x''') \implies \|\omega'' = \|\omega'''.$$

This statement will be proved now.

For all letters $x \in [\omega'']$, let $\|_x x = \|_x x' = \|_x x'' = \|_x x'''$. If (q, y) is an edge of any $\|x^{(i)}$, then (q, y) is also an edge of $\|x^{(i+1)}$ (because of $\|_y y = \|_y y'$). Hence, each edge set includes those of lower derivatives, that is $\|x \subseteq \|x' \subseteq \|x'' \subseteq \|x'''$.

In order to conclude that $\|\omega'' = \|\omega'''$, the inclusion $\|x'' \subseteq \|x'''$ must be an equation. In the sequel, the inverted inclusion ($\|x''' \subseteq \|x''$) will be shown. The case distinction used follows from Proposition 6.

1. $[x'] = \{x\}$, hence $[x^{(n)}] = \{x\}$ for all natural numbers n . Especially, x'' is equal to x''' , and also $\|x'' = \|x'''$.
2. $[x'] = \{x, \bar{x}\}$.
 - (a) $[x''] = [x']$, hence $[x^{(n)}] = [x']$ for all natural numbers n . Because of the second synchronization condition, the letters x and \bar{x} alternate in x'' and in x''' . That means for the edge sets

$$\begin{aligned} \|x'' &= \|x \cup \|x^{(o)} \bar{x} \cup \|x \cup \|x^{(o)} \bar{x} \cup \dots \|x & (\text{Prop. 5}) \\ &= \|x'''. \end{aligned}$$

- (b) $[x''] = \mathcal{A}$. The edge set $\|x'$ consists of the edges (o, x) and $(x(o), \bar{x})$ (due to the second synchronization condition). The edge set $\|x''$ does not contain any other x -edge nor \bar{x} -edge (the edge (o, x) can produce

the edges of $\|x'$ only; the edge $(x(o), \bar{x})$ can produce (o, x) - due to $\|_x x = \|_x x''$ - and $(x(o), \bar{x})$ - due to $\|_{\bar{x}} \bar{x} = \|_{\bar{x}} \bar{x}'$ - only). Hence, the x^\perp - and \bar{x}^\perp -edges occur pairwise (between the same points):





$$(q, x^\perp) \in \|x'' \iff (x^\perp(q), \bar{x}^\perp) \in \|x''.$$

The set $\|_x x'''$ consists of (o, x) only. The set $\|_{\bar{x}} \bar{x}'''$ consists of $(x(o), \bar{x})$ only (due to $\|_{\bar{x}} \bar{x} = \|_{\bar{x}} \bar{x}''$). The x^\perp - and \bar{x}^\perp -edges do not produce new x -, \bar{x} -edges (because of the same reasons), thus, the x^\perp - and \bar{x}^\perp -edges occur pairwise in $\|x'''$; x^\perp does not produce a new x^\perp -edge, hence, nor a new \bar{x}^\perp -edge. Because of the same reason for \bar{x}^\perp , there is not an edge in $\|x'''$ which is not in $\|x''$.

$$3. [x'] = \{x, x^\perp, \bar{x}^\perp\}.$$

(a) $[x''] = [x']$, hence $[x^{(n)}] = [x']$ for all natural numbers n . As there are no \bar{x} -edges in the edge sets of any derivative, the edge set does not change from the first to the second derivative: $\|x'' = \|x'$. Because of Proposition 10, it follows $\|x'' = \|x'''$.





(b) $[x''] = \mathcal{A}$. This means, \bar{x} occurs in the derivative of x^\perp or \bar{x}^\perp . As above, the x^\perp - and \bar{x}^\perp -edges occur pairwise in $\|x'$. With each \bar{x} -edge, also an x -edge is produced. If there are more than one x^\perp - and \bar{x}^\perp -edges, then they have different positions with respect to the edge (o, x) and then another x -edge will appear. This is a contradiction. Hence, there are one x^\perp - and one \bar{x}^\perp -edge only. The following cases are the only possibilities (the graphs given to the right shall illustrate the case of $x = r$):

- i. $\|x' = \{(o, x^\perp), (x^\perp(o), \bar{x}^\perp), (o, x)\}$ 
- ii. $\|x' = \{(o, x), (x(o), x^\perp), ((xx^\perp)(o), \bar{x}^\perp)\}$ 
- iii. $\|x' = \{(o, \bar{x}^\perp), (\bar{x}^\perp(o), x^\perp), (o, x)\}$ 
- iv. $\|x' = \{(o, x), (x(o), \bar{x}^\perp), ((x\bar{x}^\perp)(o), x^\perp)\}$ 

In $\|x''$, there are not any new x -, x^\perp -, \bar{x}^\perp -edges. Hence, the new \bar{x} -edge is $(x(o), \bar{x})$ (but then $\|x'''$ is the same as $\|x''$, because it does not get any new edge), or the new \bar{x} -edge is $((xx^\perp)(o), \bar{x})$ (in the cases i. and ii.) or $((x\bar{x}^\perp)(o), \bar{x})$ (in the cases iii. and iv.). Then a new vertex ϵ appears:

- i. $\epsilon = (xx^\perp)(o)$ 
- ii. $\epsilon = x^\perp(o)$ 
- iii. $\epsilon = (x\bar{x}^\perp)(o)$ 
- iv. $\epsilon = \bar{x}^\perp(o)$ 

Since ϵ is neither o nor $x(o)$, there is another edge in $\|x''$ that is not in the set $\|x'$:

- i. $(x(o), x^\perp) \in \|x''$ 
- ii. $(x^\perp(o), \bar{x}^\perp) \in \|x''$ 
- iii. $(x(o), \bar{x}^\perp) \in \|x''$ 
- iv. $(\bar{x}^\perp(o), x^\perp) \in \|x''$ 

This implies that the \bar{x} -edge is produced by \bar{x}^\perp in the cases i. and iv., and by x^\perp in the other cases. Both do not produce further \bar{x} -edges (otherwise they would produce a new x -edge or new x^\perp -, \bar{x}^\perp -edges). As the \bar{x} -edge cannot produce any new edge, the set $\|x'''$ has no additional edges: $\|x''' \subseteq \|x''$.

4. $[x'] = \mathcal{A}$. Let v, w, y be different letters of the set $\mathcal{A} \setminus \{x\}$. The x - and v -edges produced by deriving v in x' are not new (they are in $\|x'$ already). Possibly, a new w -edge occurs. The x -, v -, and w -edges produced by deriving w in x'' are in $\|x''$ already. Similar to the previous case, also all arising y -edges are in $\|x''$. Hence, all edges of $\|x'''$ are elements of $\|x''$.

There are no other cases (s. Prop. 6). Every case yields that $\|x''' \subseteq \|x''$. Together with the inclusion $\|x'' \subseteq \|x'''$, one obtains that $\|x'' = \|x'''$. Hence, the edge sets, with respect to any point $a \in \mathbb{Z}^2$, coincide: $\|a x'' = \|a x'''$. Let w_1, \dots, w_n be the letters of ω : $\omega = w_1 \dots w_n$. For the edge set of the second derivative of ω , the considerations above yield

$$\begin{aligned}
 \|\omega'' &= \|w_1'' \cup \|w_1''(o)w_2'' \cup \dots \cup \|\overrightarrow{w_{n-1}''}(o)w_n'' & (\text{Prop. 5}) \\
 &= \|w_1'' \cup \|w_1''(o)w_2'' \cup \dots \cup \|\overrightarrow{w_{n-1}'''}(o)w_n'' & (\text{Prop. 8}) \\
 &= \|w_1''' \cup \|w_1'''(o)w_2''' \cup \dots \cup \|\overrightarrow{w_{n-1}'''}(o)w_n''' & (\|a x'' = \|a x''') \\
 &= \|\omega''' & (\text{Prop. 5}).
 \end{aligned}$$

This proves the proposition. \square

In the sequel, consider ω such that $\|_x x \neq \|_x x^{(l)}$ for a letter $x \in [\omega'']$ and a derivation step $l \in \{1, 2, 3\}$. The edge set $\|x$ consists of the edge (o, x) . Another x -edge is in the graph of the l -th derivative of x : $(q, x) \in \|x^{(l)}$ with $q \neq o$. According to Proposition 9, one obtains that $\|q x^{(l)} \subseteq \|x^{(2l)}$, which means that all edges of the l -th derivative are in the $2l$ -th derivative displaced by the point q . Thus, it is especially $(q + q, x) \in \|x^{(2l)}$. Induction leads to $(nq, x) \in \|x^{(nl)}$ in general.

Consider $\omega'' = vx\bar{v}$, then the edge sets of every l -th derivative of ω'' are

$$\begin{aligned}
 \|\omega^{(nl+2)} &= \|\bar{v}^{(nl)} \cup \|\bar{v}^{(nl)}(o)x^{(nl)} \cup \|(\bar{v}x)^{(nl)}(o)\bar{v}^{(nl)} \\
 &= \|\bar{v}^{(nl)} \cup \|\bar{v}^{(nl)}(o)x^{(nl)} \cup \|(\bar{v}x)^{(nl)}(o)\bar{v}^{(nl)} & (\text{Prop. 8}).
 \end{aligned}$$

The edge $(nq + v(o), x)$ occurs in every set $\|\bar{v}^{(nl)}x^{(nl)}$ with $n \in \mathbb{N}_0$. Hence, for each $n \in \mathbb{N}_0$, the edge $(nq + v(o), x)$ is also an element of the set $\|\omega^{(nl+2)}$. Thus, the vertex $nq + v(o)$ is an element of the vertex set $\odot(\omega^{(nl+2)})$.

Let X_ω be the set of all vertices $nq + v(o)$, V_ω the union of the vertex sets $\odot(\omega^{(nl+2)})$, and P_ω the set of all pictures $p(\omega^{(nl+2)})$ over all $n \in \mathbb{N}_0$:

$$\begin{aligned} X_\omega &= \{ nq + v(o) \mid n \in \mathbb{N}_0 \}, \\ V_\omega &= \bigcup_{n \in \mathbb{N}_0} \odot(\omega^{(nl+2)}), \\ P_\omega &= \{ p(\omega^{(nl+2)}) \mid n \in \mathbb{N}_0 \}. \end{aligned}$$

The set X_ω is infinite because $q \neq o$. Each point in X_ω also occurs in V_ω ; thus V_ω is also infinite. Every vertex set $\odot(\omega^{(nl+2)})$ with $n \in \mathbb{N}_0$ is finite; hence, there are infinitely many different ones in the union V_ω . If the vertex sets of two strings $x, y \in \mathcal{A}^*$ differ, then the pictures are also different. Hence, P_ω contains infinitely many different pictures. This result is summarized in the next proposition.

Proposition 12. *If there exists a letter $x \in [\omega'']$ such that one of the first three derivation edge sets $\|x'$, $\|x''$ or $\|x'''$ contains an edge different from (o, x) , then the picture language generated is infinite.*

Together with the Proposition 11, this leads immediately to the following lemma.

Lemma 2. *Let $G = (\mathcal{A}, h, \omega)$ be a length-constant sDOL system. If the edge sets of the second and third derivatives of the axiom ω do not coincide, then the picture language generated is infinite.*

Hence, the second supposition on page 65 is confirmed and proved. The Lemmas 1 and 2 together state that the difference of the edge sets of the second and third derivatives of the axiom is a necessary and sufficient criterion of the finiteness. This result is summarized in the next theorem.

Theorem 3. *Let $G = (\mathcal{A}, h, \omega)$ be a length-constant sDOL system. The picture language P_G generated by G is finite if and only if the edge sets of the second and third derivatives of the axiom ω coincide.*

From an algorithmic point of view, the case $\mu = 0$ must be emphasized.

According to the second synchronization condition, the edge set $\|x^{(n)}$ of each derivative of any letter $x \in \mathcal{A}$ with respect to the point $a \in \mathbb{Z}^2$ consists of the edge $(a, a + v_x)$. Hence, the pictures of all derivatives of ω coincide (Prop. 4).

Theorem 4. *Let $G = (\mathcal{A}, h, \omega)$ be an sDOL system with a $(1, 0)$ -endomorphism h . Then the picture language P_G generated by G is a singleton set, that is*

$$P_G = \{ p(\omega) \}.$$

This means that the finiteness can be stated immediately in the case $\mu = 0$ (without further investigation of the system).

4 Conclusion and Future Work

The paper investigates synchronous deterministic Chain Code Picture Systems with respect to the finiteness of their picture languages.

Let $G = (\mathcal{A}, h, \omega)$ be an *sDOL* system with a (κ, μ) -endomorphism h . The synchronization parameter κ defines a division of the *sDOL* systems in length-contracting ($\kappa < 1$), length-constant ($\kappa = 1$), and length-expanding ($\kappa > 1$) systems.

The following table summarizes the results:

$$\begin{aligned}
 \kappa < 1: \quad & P_G = \{ p(\omega), p(\omega'), p(\omega''), p(\omega''') \} \\
 \kappa > 1: \quad & |P_G| = \infty \\
 \kappa = 1: \quad & \mu = 0 \implies P_G = \{ p(\omega) \} \\
 & s(\omega'') = s(\omega''') \implies P_G = \{ p(\omega), p(\omega'), p(\omega'') \} \\
 & s(\omega'') \neq s(\omega''') \implies |P_G| = \infty
 \end{aligned}$$

If the picture language generated by an *sDOL* system is finite, then it consists of four elements at most. In addition, an algorithm is given that decides for any *sDOL* system G , whether the picture language P_G generated is finite or not.

The decision about the finiteness of the picture language of a given *sDOL* system $G = (\mathcal{A}, h, \omega)$ can be made immediately (without further investigation of the system) if the (κ, μ) -endomorphism h is length-contracting or length-expanding or length-constant with $\mu = 0$. Otherwise, the start string must be derived three times, and the edge sets of the second and third derivatives must be checked for equality. This time effort is cubic in the lengths of the replacement strings.

For synchronous deterministic Chain Code Picture Systems $G = (\mathcal{A}, h, \omega)$, the finiteness and infiniteness are decidable in time $\mathcal{O}(pn^3)$, where $p = \#\omega$ is the length of the start string ω and $n = \max \{ \#h(x) \mid x \in \mathcal{A} \}$ is the maximum length of the replacement strings.

Future work will address the finiteness of picture languages generated by non-deterministic *sOL* systems and tabled systems such as *sDTOL* systems and *sTOL* systems. The deterministic systems generate four pictures at most in the case of finiteness. For applications however, systems are desirable that generate a large but a finite picture language.

Acknowledgements

The author would like to thank Prof. Hollatz for many stimulating discussions and recommendations, Prof. Dassow for his support and the anonymous referees for their critical remarks and helpful suggestions.

References

- [DH89] DASSOW, J.; HINZ, F.: *Kettenkode-Bildsprachen. Theorie und Anwendungen*. Wiss. Zeitschrift d. Techn. Univ. Magdeburg, 33, 1989.
- [DHr92] DASSOW, J.; HROMKOVIČ, J.: *On Synchronized Lindenmayer Picture Languages*. *Lindenmayer Systems*, 253–261. Springer-Verlag, Berlin 1992.
- [Fre74] FREEMAN, H.: *Computer processing of line-drawing images*. *Computer Surveys*, 6:57–97, 1974.

Received December, 2000

Quasi-star-free Languages on Infinite Words*

Zhilin Wu[†]

Abstract

Quasi-star-free languages were first introduced and studied by Barrington, Compton, Straubing and Thérien within the context of circuit complexity in 1992, and their connections with propositional linear temporal logic were established by Ésik and Ito recently. While these results are all for finite words, in this paper we consider the languages on infinite words.

1 Introduction

Characterizations of different subclasses of regular languages have been a constantly active research area since Büchi characterized regular languages by monadic second order logic in [3]. One of the most important characterizations among them is the characterization of star free languages: in [11, 17, 9, 7, 13, 19, 18, 4], star free languages on finite and infinite words were characterized by aperiodic monoids, monadic first order logic and linear temporal logic.

Quasi-star-free languages were first studied by Barrington, Compton, Straubing and Thérien in [2]. Their motivation was to characterize the regular languages that can be recognized by constant-depth Boolean circuits using OR, AND and NOT gates (AC^0). They found that these languages are precisely the quasi-star-free languages. And they give a characterization in terms of quasi-aperiodic semigroups and in terms of first order logic $FO[C]$ which uses only the numerical predicates $x < y$ and $x \equiv r \pmod{d}$. Recently, Ésik and Ito proved in [5] that $FO[C]$ and propositional linear temporal logic with cyclic counting ($LTL[C]$) have the same expressive power. While these results are all for finite words, we extend them to the case of infinite words in this paper.

This paper is organized as follows. In section 2 we give some preliminaries about regular languages on finite and infinite words. Then in section 3, we give some definitions of quasi-star-free languages on finite words (QSF^F), and summarize

*Partially supported by the National Natural Science Foundation of China under Grant No. 60223005 and the National Grand Fundamental Research 973 Program of China under Grant No. 2002cb312200.

[†]Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, P.O.Box 8718, Beijing, China, & Graduate School of the Chinese Academy of Sciences, 19 Yuquan Street, Beijing, China. E-mail: wuzl@ios.ac.cn

the results of QSF^F in [2, 5]. In section 4, we define quasi-star-free languages on infinite words (QSF^I), and extend the results of QSF^F to QSF^I . Finally in section 5, we give some conclusions and remarks on this paper.

2 Preliminaries

2.1 Regular languages on finite words

In this subsection, at first we present some basic facts of semigroups and formal languages on finite words (cf. [12, 6, 14, 10] for more information), then after recalling the definitions of monadic first order logic ($\text{FO}[<]$) and linear temporal logic (LTL) interpreted on finite words, we introduce the classical results of star free languages on finite words.

Let A be a finite alphabet, and $L \subseteq A^*$ be regular.

2.1.1 Monoids and formal languages on finite words

Let M be a finite monoid. We say that morphism $\phi : A^* \rightarrow M$ recognizes L if there is $X \subseteq M$ such that $L = X\phi^{-1}$. And we say that monoid M recognizes L if there is a morphism $\phi : A^* \rightarrow M$ recognizing L . Moreover we say that congruence \approx on A^* recognizes L if the natural morphism $\phi : A^* \rightarrow A^*/\approx$ recognizes L .

The syntactic congruence of L , \approx_L , is defined by: $u \approx_L v$ iff $(xuy \in L \text{ iff } xvy \in L \text{ for all } x, y \in A^*)$; the syntactic monoid of L , $M(L)$, is defined by the quotient monoid A^*/\approx_L ; and the syntactic morphism of L , $\eta_L : A^* \rightarrow M(L)$, is defined by $u\eta_L = [u]$, where $[u]$ denotes the equivalence class of \approx_L containing u . Syntactic congruence is the coarsest congruence of A^* recognizing L , i.e. for any congruence \approx recognizing L , $u \approx v$ implies $u \approx_L v$ for all $u, v \in A^*$.

A morphism $\phi : A^* \rightarrow M$ recognizes L iff there is a morphism $\theta : \text{Im}(\phi) \rightarrow M(L)$ (where $\text{Im}(\phi)$ is the image of ϕ) such that for all $u \in A^*$, $u(\phi\theta) = u\eta_L$. Furthermore, a morphism $\phi : A^* \rightarrow M$ recognizes L iff there are morphisms $\phi' : A^* \rightarrow M'$ and $\theta : \text{Im}(\phi) \rightarrow M'$ such that ϕ' recognizes L and for all $u \in A^*$, $u(\phi\theta) = u\phi'$.

L is star free if L can be constructed from singleton languages $\{a\} (a \in A)$ and the language A^* by finite applications of operations of union, complementation, and concatenation.

L is noncounting if there is some $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$, $xy^n z \in L$ iff $xy^{n+1} z \in L$ for all $x, y, z \in A^*$.

A monoid M is aperiodic if there is some $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$, $m^n = m^{n+1}$ for all $m \in M$.

L is aperiodic if $M(L)$ is aperiodic. It is easy to show that L is aperiodic iff there is an aperiodic monoid M recognizing L .

It is not hard to show that L is noncounting iff L is aperiodic. In the remainder of this paper, we don't distinguish between the "noncounting" and "aperiodic" properties of regular languages on finite words.

2.1.2 First order logic and linear temporal logic on finite words

Let $\text{FO}[<]$ denote first order logic on words with binary predicate $<$ and unary predicates $P_a (a \in A)$. The formulas of $\text{FO}[<]$ are defined by the following rules:

$$\varphi := P_a(x) \mid x < y \mid \varphi_1 \vee \varphi_2 \mid \neg\psi \mid (\exists x)\psi$$

The semantics of $\text{FO}[<]$ are defined as follows: let X be a variable set and φ be a formula with free variables in X ; $u \in A^*$ and $\eta : X \rightarrow \{0, \dots, |u|\}$, i.e., η maps variables in X to “positions” in u .

- $(u, \eta) \models P_a(x)$, if $u[\eta(x)] = a$, where $u[\eta(x)]$ is the letter of u at position $\eta(x)$ (the first position is 0, the last position is $|u|$, and by convention the letter at position $|u|$ is ε);
- $(u, \eta) \models x < y$, if $\eta(x) < \eta(y)$;
- $(u, \eta) \models \varphi_1 \vee \varphi_2$, if $(u, \eta) \models \varphi_1$ or $(u, \eta) \models \varphi_2$;
- $(u, \eta) \models \neg\psi$, if not $(u, \eta) \models \psi$;
- $(u, \eta) \models (\exists x)\psi$, if there exists a function $\eta' : X \rightarrow \{0, \dots, |u|\}$, which agrees with η on $X - \{x\}$ and possibly differs from η on x , such that $(u, \eta') \models \psi$.

Let φ be an $\text{FO}[<]$ sentence and $u \in A^*$. We write $u \models \varphi$ if there is an $\eta : X \rightarrow \{0, \dots, |u|\}$ such that $(u, \eta) \models \varphi$.

Remark 2.1. *The semantics of $\text{FO}[<]$ defined in [5] had a subtle inaccuracy: the assignments of variables were defined by function $\lambda : X \rightarrow [|u|]$, where $[|u|] = \{0, \dots, |u| - 1\}$. But then for the empty string ε , the assignments would become into $\lambda : X \rightarrow \emptyset$, since $[|\varepsilon|] = [0] = \emptyset$.*

We avoid the accuracy by defining the assignments as $\eta : X \rightarrow \{0, \dots, |u|\}$, and thus formulas of $\text{FO}[<]$ can be interpreted on the empty string ε .

It is natural to define the boolean operations “ \wedge ”, “ \rightarrow ”, etc. in a standard way. Here we introduce several other abbreviations for $\text{FO}[<]$: *Last*(x) for $\forall y (\neg(x < y))$; *True* for $\varphi \vee \neg\varphi$, where φ is a fixed sentence; and *False* for $\neg\text{True}$.

A language $L \subseteq A^*$ is definable in $\text{FO}[<]$ if there is an $\text{FO}[<]$ sentence φ such that for all $u \in A^*$, $u \models \varphi$ iff $u \in L$.

Associate each letter a in A with a propositional constant p_a . Then formulas of linear temporal logic (LTL, [15]) over alphabet A are defined by the following rules:

$$\varphi := p_a \mid \varphi_1 \vee \varphi_2 \mid \neg\psi \mid X\psi \mid \varphi_1 U \varphi_2$$

The semantics of LTL formulas on finite words are defined as follows: Let φ be an LTL formula, $u \in A^*$. Denote the suffix of u starting from the i -th position (the first position is 0) as u^i , where $0 \leq i \leq |u|$, and the suffix starting from the $|u|$ -th position is empty string ε .

- $u \models p_a$, if $u = av$, for some $v \in A^*$;
- $u \models \varphi_1 \vee \varphi_2$, if $u \models \varphi_1$ or $u \models \varphi_2$;
- $u \models \neg\varphi_1$, if not $u \models \varphi_1$;
- $u \models X\varphi_1$, if $|u| > 0$ and $u^1 \models \varphi_1$;
- $u \models \varphi_1 U \varphi_2$, if there is $0 \leq i \leq |u|$ such that $u^i \models \varphi_2$ and for all $0 \leq j < i$, $u^j \models \varphi_1$.

We introduce several abbreviations for LTL, let $True \equiv p_a \vee \neg p_a$, where a is any letter in A , and let $False \equiv \neg True$. Moreover, let End denote the formula $\bigwedge_{a \in A} \neg p_a$, so that for all $u \in A^*$, $u \models End$ iff $u = \varepsilon$.

Remark 2.2. *When interpreted on finite words, the LTL formulas $\neg X\varphi$ and $X\neg\varphi$ are not equivalent while on infinite words they are (See Section 2.2.2 for LTL interpreted on infinite words). For instance, $\varepsilon \models \neg Xp_a$ while not $\varepsilon \models X\neg p_a$, where ε is the empty string.*

A language $L \subseteq A^*$ is LTL definable iff there is an LTL formula φ such that for all $u \in A^*$, $u \models \varphi$ iff $u \in L$.

2.1.3 Classical results of star free languages on finite words

The classical results of star free languages on finite words are summarized in the following proposition:

Proposition 2.3. *Let $L \subseteq A^*$ be regular. The following conditions are equivalent [11, 17, 9, 7, 4]:*

- L is star free;
- L is aperiodic;
- $M(L)$ contains no nontrivial group (i.e. contains no subsets which form a nontrivial group under the product of $M(L)$);
- L is $FO[<]$ definable;
- L is LTL definable.

2.2 Regular languages on infinite words

Similar to the case of finite words, in this subsection at first we present some basic facts of semigroup and formal languages on infinite words (cf. [1, 20, 21, 4, 16]), then we interpret monadic first order logic ($FO[<]$) and linear temporal logic (LTL) on infinite words, at last we introduce the classical results of star free languages on infinite words.

Let A be a finite alphabet and $L \subseteq A^\omega$ be regular, i.e., $L = \bigcup_{i=1}^m X_i Y_i^\omega$, where $X_i \subseteq A^*$, $Y_i \subseteq A^+$ are regular languages on finite words.

2.2.1 Monoids and formal languages on infinite words

Let M be a finite monoid. L is recognized by morphism $\phi : A^* \rightarrow M$ if for all $m, n \in M$, $(m\phi^{-1})(n\phi^{-1})^\omega \cap L \neq \emptyset$ implies $(m\phi^{-1})(n\phi^{-1})^\omega \subseteq L$. A monoid M recognizes L iff there is a morphism $\phi : A^* \rightarrow M$ recognizing L . Moreover we say that a congruence \approx on A^* recognizes L if the natural morphism $\phi : A^* \rightarrow A^*/\approx$ recognizes L .

The syntactic congruence of L , \approx_L , is defined by: for all $u, v \in A^*$, $u \approx_L v$ iff for all $x, y, z \in A^*$, $(xuyz^\omega \in L \text{ iff } xvyz^\omega \in L)$ and $(x(yuz)^\omega \in L \text{ iff } x(yvz)^\omega \in L)$. The syntactic monoid of L , $M(L)$, is defined by the quotient monoid A^*/\approx_L . The syntactic morphism of L , $\eta_L : A^* \rightarrow M(L)$, is defined by $u\eta_L = [u]$, where $[u]$ is the equivalence class of \approx_L containing u . Syntactic congruence is the coarsest congruence recognizing L .

Proposition 2.4. *Let $L \subseteq A^\omega$ be regular. A morphism $\phi : A^* \rightarrow M$ recognizes L iff there is a morphism $\theta : Im(\phi) \rightarrow M(L)$ such that for all $u \in A^*$, $u\phi\theta = u\eta_L$.*

Proof.

" \Rightarrow " part:

Define $\theta : Im(\phi) \rightarrow M(L)$ as follows:

$$m\theta = u\eta_L, \text{ where } u \in A^*, u\phi = m$$

θ is well defined since $u\phi = v\phi$ implies that $u\eta_L = v\eta_L$ (syntactic congruence is the coarsest one).

It is easy to verify that $\phi\theta = \eta_L$

" \Leftarrow " part:

It is sufficient to prove that for all $m, n \in Im(\phi)$

$$\phi^{-1}(m)[\phi^{-1}(n)]^\omega \cap L \neq \emptyset \text{ implies } \phi^{-1}(m)[\phi^{-1}(n)]^\omega \subseteq L$$

Since $\phi^{-1}(m)[\phi^{-1}(n)]^\omega \cap L$ is a nonempty regular language, there is an ultimately periodic ω -word $xy^\omega \in \phi^{-1}(m)[\phi^{-1}(n)]^\omega \cap L$. So xy^ω has a decomposition: $w_0w_1^\omega$ such that

$$w_0 \in \phi^{-1}(m)[\phi^{-1}(n)]^p, w_1 \in [\phi^{-1}(n)]^q \text{ for some } p, q \geq 0$$

It is easy to see that $\phi^{-1}(m)[\phi^{-1}(n)]^\omega \subseteq [w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega$, thus it is sufficient to prove that $[w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega \subseteq L$, i.e., $[w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega \cap \bar{L} = \emptyset$.

To the contrary, suppose that $[w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega \cap \bar{L} \neq \emptyset$.

Since $[w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega \cap \bar{L}$ is regular, then there is an ultimately periodic word $\alpha_0\alpha_1^\omega \in [w_0\phi\phi^{-1}][w_1\phi\phi^{-1}]^\omega \cap \bar{L}$.

$\alpha_0\alpha_1^\omega$ has a decomposition $\alpha'_0\alpha'_1^\omega$ such that $\alpha'_0 \in w_0\phi\phi^{-1}[w_1\phi\phi^{-1}]^r$ and $\alpha'_1 \in [w_1\phi\phi^{-1}]^s$ for some $r, s \geq 0$.

From the assumption $\phi\theta = \eta_L$ we know that $\alpha'_0\eta_L = \alpha'_0\phi\theta = (w_0w_1^r)\phi\theta = (w_0w_1^r)\eta_L$, and $\alpha'_1\eta_L = \alpha'_1\phi\theta = (w_1^s)\phi\theta = (w_1^s)\eta_L$. Thus $w_0w_1^r(w_1^s)^\omega \in L$ iff $\alpha \in L$, i.e., $w_0w_1^\omega \in L$ iff $\alpha \in L$, i.e., $xy^\omega \in L$ iff $\alpha \in L$, a contradiction. \square

Corollary 2.5. *A morphism $\phi : A^* \rightarrow M$ recognizes L iff there are morphisms $\phi' : A^* \rightarrow M'$ and $\theta : \text{Im}(\phi) \rightarrow M'$ such that ϕ' recognizes L and for all $u \in A^*$, $u(\phi\theta) = u\phi'$.*

L is star free if L can be constructed from the language A^ω by finite applications of operations of union, complementation and concatenation on the left by star free languages of A^* .

L is noncounting if there is $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and $x, u, y, z \in A^*$, $(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L)$ and $(x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L)$.

L is aperiodic if its syntactic monoid $M(L)$ is aperiodic. And it is easy to show that L is aperiodic iff it is recognized by an aperiodic monoid.

It is not hard to prove that L is noncounting iff L is aperiodic. In the remainder of this paper, for regular languages on infinite words, we don't distinguish between the "noncounting" and "aperiodic" properties.

2.2.2 First order logic and linear temporal logic on infinite words

FO[<] and LTL formulas can also be interpreted on infinite words.

For FO[<]: Let X be the variable set and φ be a formula with free variables in X ; $u \in A^\omega$ and $\eta : X \rightarrow \mathbb{N}$, i.e., η maps variables in X to "positions" in u .

- $(u, \eta) \models P_a(x)$, if $u[\eta(x)] = a$, where $u[\eta(x)]$ is the $\eta(x)$ th letter of u ;
- $(u, \eta) \models x < y$, if $\eta(x) < \eta(y)$;
- $(u, \eta) \models \varphi_1 \vee \varphi_2$, if $(u, \eta) \models \varphi_1$ or $(u, \eta) \models \varphi_2$;
- $(u, \eta) \models \neg\psi$, if not $(u, \eta) \models \psi$;
- $(u, \eta) \models (\exists x)\psi$, if there exists a function $\eta' : X \rightarrow \mathbb{N}$, which agrees with η on $X - \{x\}$ and possibly differs from η on x , such that $(u, \eta') \models \psi$.

Let φ be an FO[<] sentence and $u \in A^\omega$. We write $u \models \varphi$ if there is an $\eta : X \rightarrow \mathbb{N}$ such that $(u, \eta) \models \varphi$.

For LTL: Let φ be an LTL formula, $u \in A^\omega$. Denote the suffix of u starting from i -th position (the first position is 0) as u^i , then

- $u \models p_a$, if $u = av$, for some $v \in A^\omega$;
- $u \models \varphi_1 \vee \varphi_2$, if $u \models \varphi_1$ or $u \models \varphi_2$;
- $u \models \neg\varphi_1$, if not $u \models \varphi_1$;
- $u \models X\varphi_1$, if $u^1 \models \varphi_1$;
- $u \models \varphi_1 U \varphi_2$, if there is $i \geq 0$ such that $u^i \models \varphi_2$ and for all $0 \leq j < i$, $u^j \models \varphi_1$.

L is definable in FO[<] if there is an FO[<] sentence φ such that for all $u \in A^\omega$, $u \models \varphi$ iff $u \in L$.

L is definable in LTL if there is an LTL formula φ such that for all $u \in A^\omega$, $u \models \varphi$ iff $u \in L$.

2.2.3 Classical results of star free languages on infinite words

Similar to the finite words, there are the following classical results of star free languages on infinite words.

Proposition 2.6. *Let $L \subseteq A^\omega$ be regular. The following conditions are equivalent [13, 19, 18, 9, 7]:*

- L is star free;
- L is aperiodic;
- $M(L)$ contains no nontrivial group;
- $L = \bigcup_{i=1}^m X_i Y_i^\omega$, where $X_i \subseteq A^*$, $Y_i \subseteq A^+$ are star free and $Y_i Y_i \subseteq Y_i$;
- L is $FO[<]$ definable;
- L is LTL definable.

3 Quasi-star-free languages on finite words

3.1 Quasi-star-free languages on finite words

Definition 3.1. *Let $L \subseteq A^*$ be regular. L is quasi-star-free if there is some $d \geq 1$ such that L can be constructed from singleton languages $\{a\} (a \in A)$ and the language $(A^d)^*$ by finite applications of operations of union, complementation, and concatenation.*

If $L \subseteq A^*$ is star free, it is quasi-star free as well.

The family of quasi-star-free languages on finite words is denoted by QSF^F .

Definition 3.2. *Let $L \subseteq A^*$ be regular. L is quasi-noncounting if there is some $d \geq 1$ such that there is some $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$, and for all $x, y, z \in A^*$ with $|y| = 0 \bmod d$; $xy^n z \in L$ iff $xy^{n+1} z \in L$.*

Let $L \subseteq A^*$ be regular and $\eta_L : A^* \rightarrow M(L)$ be its syntactic morphism. we denote $(A^d)^* \eta_L$ by $M(L)^{(d)}$. Then we have the following definition:

Definition 3.3. *Let $L \subseteq A^*$ be regular and $\eta_L : A^* \rightarrow M(L)$ be its syntactic morphism. L is quasi-aperiodic if there is $d \geq 1$ such that $M(L)^{(d)}$ is aperiodic.*

A language of A^* is quasi-noncounting iff it is quasi-aperiodic. Thus in the remainder of this paper, we don't distinguish between the "quasi-noncounting" and "quasi-aperiodic" properties of regular languages on finite words.

3.2 Logic with cyclic counting interpreted on finite words

$\text{FO}[\prec]$ can be extended with unary predicates $C_d^r(d \geq 1, 0 \leq r < d)$ adjoined. C_d^r are interpreted on finite words as follows:

Let $u \in A^*$, $\eta : X \rightarrow \{0, \dots, |u|\}$, then $(u, \eta) \models C_d^r(x)$ if $x\eta \equiv r \pmod{d}$.

Denote this extended logic of $\text{FO}[\prec]$ as $\text{FO}[C]$.

LTL can be extended with “ U ” (Until) operator of LTL replaced by new “Until” operators with cyclic counting, namely $U^{(d,r)}$ for all $d \geq 1$ and $0 \leq r < d$. The semantics of $\varphi_1 U^{(d,r)} \varphi_2$ is defined as follows:

Let $u \in A^*$, then $u \models \varphi_1 U^{(d,r)} \varphi_2$ if there is i such that $0 \leq i \leq |u|$, $i \equiv r \pmod{d}$ and $u^i \models \varphi_2$; moreover, for all j such that $(0 \leq j < i \text{ and } j \equiv r \pmod{d})$, $u^j \models \varphi_1$.

Denote this extended LTL by $\text{LTL}[C]$.

Similar to $\text{FO}[\prec]$ and LTL, we can define the languages defined by $\text{FO}[C]$ sentences and $\text{LTL}[C]$ formulas.

The expressive power of $\text{FO}[C]$ is strictly stronger than that of $\text{FO}[\prec]$. For instance, language $(\{a\}A)^*$ ($a \in A$ and $|A| > 1$) isn't aperiodic, then according to Proposition 2.3, it can't be defined in $\text{FO}[\prec]$, while it can be defined by $\text{FO}[C]$ sentence $\forall x (Last(x) \rightarrow C_2^0(x)) \wedge \forall x (C_2^0(x) \wedge \neg Last(x) \rightarrow P_a(x))$.

It is obvious that for $u \in A^*$, $u \models \varphi_1 U \varphi_2$ iff $u \models \varphi_1 U^{(1,0)} \varphi_2$. Then the expressive power of $\text{LTL}[C]$ is at least as strong as that of LTL. In fact, $\text{LTL}[C]$ is more expressive than LTL. For instance, language $(\{a\}A)^*$ ($\{a\} \in A$ and $|A| > 1$) can't be defined in LTL, while it can be defined by $\text{LTL}[C]$ formula $p_a U^{(2,0)} End$.

Remark 3.4. In [5], $\text{LTL}[C]$ is defined by adjoining additional constants $Ig_{d,r}$ ($d \geq 1, 0 \leq r < d$) into LTL, and $U^{(d,r)}$ are just derived temporal operators of $Ig_{d,r}$ and “ U ”. Nevertheless, since $u \models Ig_{d,r}$ iff $|u| \equiv r \pmod{d}$, $\text{LTL}[C]$ defined in [5] can't be interpreted on infinite words. Consequently we directly adjoin $U^{(d,r)}$ into LTL since $U^{(d,r)}$ can be interpreted on infinite words naturally. When interpreted on finite words, $Ig_{d,r}$ can be derived from $U^{(d,r)}$ as follows:

$$Ig_{d,r} \equiv True U^{(d,r)} End$$

3.3 Theorem on quasi-star-free languages on finite words

We summarize the results of quasi-star-free languages on finite words in [2, 5] into the following proposition:

Proposition 3.5. Let $L \subseteq A^*$ be regular. The following conditions are equivalent:

- (i) L is quasi-star-free;
- (ii) L is quasi-aperiodic;
- (iii) For all $t \geq 0$, $A^t \eta_L$ contains no nontrivial group;
- (iv) L is definable in $\text{FO}[C]$;
- (v) L is definable in $\text{LTL}[C]$.

Remark 3.6. (i), (ii), (iii) and (iv) of Proposition 3.5 were proved equivalent in [2], and (iv) and (v) were proved equivalent in [5]. As a matter of fact, (i), (iii), (iv) of Proposition 3.5 and the following condition (ii') (Theorem 3(d) in [2]), instead of (ii), were proved equivalent in [2],

(ii') L is recognized by a morphism $\psi : \{0, 1\}^* \rightarrow MwrZ_r$, where M is a finite aperiodic monoid and where the composition $\psi\pi : \{0, 1\}^* \rightarrow Z_r$ takes both 0 and 1 to the generator 1 of Z_r (see [2] for the exact meaning of (ii'))

And it is not hard to prove that (ii) and (ii') are equivalent.

4 Quasi-star-free languages on infinite words

4.1 Quasi-star-free languages on infinite words

Similar to the case of finite words, we define that an ω -language is quasi-star-free, quasi-noncounting and quasi-aperiodic in this subsection.

Definition 4.1. Let $L \subseteq A^\omega$ be regular. L is quasi-star-free if L can be constructed from the language A^ω by finite applications of operations of union, complementation, and concatenation on the left by quasi-star-free languages of A^* .

If an ω -language $L \subseteq A^\omega$ is star free, it is quasi-star-free as well. The family of quasi-star-free languages on infinite words is denoted by QSF^I .

Proposition 4.2. Let $L \subseteq A^\omega$ be quasi-star-free, then there is some $d \geq 1$ such that all those quasi-star-free languages of A^* , used in the construction of L (namely, used in the operations of left concatenation during the construction of L), can be constructed from singleton languages $\{a\} (a \in A)$ and the language $(A^d)^*$ by finite applications of operations of union, complementation and concatenation.

Proof. Let L_1, \dots, L_k be the quasi-star-free languages of A^* used in the construction of L .

Then there are $d_i (1 \leq i \leq k)$ such that $L_i (1 \leq i \leq k)$ can be constructed from singleton languages $\{a\} (a \in A)$ and the language $(A^{d_i})^*$.

Let d be the least common multiple of d_1, \dots, d_k . Then

$$(A^{d_i})^* = \bigcup_{r=0}^{d'_i-1} (A^d)^* A^{rd_i} = \bigcup_{r=0}^{d'_i-1} (A^d)^* \left(\bigcup_{a \in A} \{a\} \right)^{rd_i}, \text{ where } d'_i = \frac{d}{d_i}.$$

Consequently $L_i (1 \leq i \leq k)$ can be constructed from singleton languages $\{a\} (a \in A)$ and the language $(A^d)^*$ by finite applications of operations of union, complementation and concatenation. \square

Definition 4.3. Let $L \subseteq A^\omega$ be regular. L is quasi-noncounting if there is some $d \geq 1$ such that there is $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$ and $u, x, y, z \in A^*$ with $|u| = 0 \bmod d$, $(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L)$ and $(x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L)$.

Definition 4.4. Let $L \subseteq A^\omega$ be regular and $\eta_L : A^* \rightarrow M(L)$ be its syntactic morphism. Then L is quasi-aperiodic if there is some $d \geq 1$ such that $M(L)^{(d)}$ is aperiodic.

Proposition 4.5. Let $L \subseteq A^\omega$ be regular. L is quasi-noncounting iff it is quasi-aperiodic.

Proof.

“ \Rightarrow ” part:

Suppose that there is some $d \geq 1$ such that there is some $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$, and for all $x, u, y, z \in A^*$ with $|u| \equiv 0 \pmod{d}$;

$$(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L) \text{ and } (x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L).$$

Now we prove that $M(L)^{(d)}$ is aperiodic.

Let $m \in M(L)^{(d)}$. Then there is some $u \in (A^d)^*$ such that $u\eta_L = m$. Thus for any $n \geq n_0$, and for all $x, y, z \in A^*$;

$$(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L) \text{ and } (x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L).$$

Consequently for any $n \geq n_0$, $(u^n)\eta_L = (u^{n+1})\eta_L$, i.e., $m^n = m^{n+1}$.

“ \Leftarrow ” part:

Suppose that there is some $d \geq 1$ such that $M(L)^{(d)}$ is aperiodic, i.e., there is some $n_0 \in \mathbb{N}$ satisfying that for all $n \geq n_0$ and $m \in M(L)^{(d)}$; $m^n = m^{n+1}$.

Now we prove that L is quasi-noncounting.

Let $n \geq n_0$ and $x, u, y, z \in A^*$ with $|u| \equiv 0 \pmod{d}$. Then $u\eta_L \in M(L)^{(d)}$, so $(u^n)\eta_L = (u^{n+1})\eta_L$. From the definition of η_L , we have that

$$(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L) \text{ and } (x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L).$$

□

As a result of Proposition 4.5, in the remainder of this paper, we don't distinguish between “quasi-noncounting” and “quasi-aperiodic” properties of regular languages on infinite words.

4.2 Logic with cyclic counting interpreted on infinite words

FO[C] and LTL[C] defined in Section 3.2 can be interpreted on infinite words as follows:

For FO[C]: Let $u \in A^\omega$ and $\eta : X \rightarrow N$, then

$$(u, \eta) \models C_d^r(x) \text{ if } x\eta \equiv r \pmod{d}.$$

For LTL[C]: Let $u \in A^\omega$, then

$u \models \varphi_1 U^{(d,r)} \varphi_2$ if there is $i \geq 0$ such that $(i \equiv r \pmod{d})$ and $(u^i \models \varphi_2)$, and (for all $0 \leq j < i$ and $j \equiv r \pmod{d}$; $u^j \models \varphi_1$).

Similar to the case of finite words, we can define the languages defined by $\text{FO}[C]$ sentences and $\text{LTL}[C]$ formulas.

When interpreted on infinite words, the expressive power of $\text{FO}[C]$ ($\text{LTL}[C]$ resp.) is strictly stronger than $\text{FO}[<]$ (LTL resp.). E.g., language $(\{a\}A)^\omega$ ($a \in A$ and $|A| > 1$) isn't aperiodic, then according to Proposition 2.6, it can't be defined in $\text{FO}[<]$ (LTL resp.), while it can be defined by $\text{FO}[C]$ sentence $\forall x (C_2^0(x) \rightarrow P_a(x))$ ($\text{LTL}[C]$ formula $\neg (\text{True}U^{(2,0)}\neg p_a)$ resp.)

4.3 Theorem on quasi-star-free languages on infinite words

We extend Proposition 3.5 for QSF^F to the following theorem for QSF^I .

Theorem 4.6. *Let $L \subseteq A^\omega$ be regular. The following conditions are equivalent:*

- (i) L is quasi-star-free;
- (ii) L is quasi-aperiodic;
- (iii) For all $t \geq 0$, $A^t \eta_L \subseteq M(L)$ contains no nontrivial group;
- (iv) $L = \bigcup_{i=1}^m X_i (Y_i)^\omega$, where $X_i, Y_i \in \text{QSF}^F$, $Y_i \subseteq A^+$ and $Y_i Y_i \subseteq Y_i$;
- (v) L is definable in $\text{FO}[C]$;
- (vi) L is definable in $\text{LTL}[C]$.

Before the proof of Theorem 4.6, we give some definitions and lemmas.

Let $A^{(d)}$ denote the alphabet consisting of all letters $\langle u \rangle$, where $u \in A^d$. For any $x \in (A^{(d)})^*$, we denote the corresponding element of $(A^{(d)})^*$ as $\langle x \rangle$.

Let $L \subseteq A^*$ and $u \in A^*$, define $Lu^{-1} = \{x \mid x \in A^*, xu \in L\}$.

Let $L \subseteq A^*$ and $d \geq 1$, define

$$L^{(d)} = \begin{cases} \{\langle u_0 \rangle \dots \langle u_{k-1} \rangle \mid u_0 \dots u_{k-1} \in L, k \geq 1, \forall 0 \leq i < k (u_i \in A^d)\} & \text{if } \varepsilon \notin L \\ \{\varepsilon\} \cup \{\langle u_0 \rangle \dots \langle u_{k-1} \rangle \mid u_0 \dots u_{k-1} \in L, k \geq 1, \forall 0 \leq i < k (u_i \in A^d)\} & \text{otherwise} \end{cases}$$

Let $L \subseteq A^*$ and $u \in A^*$, define $L^{(d,u)} = (Lu^{-1})^{(d)}$.

Let $L \subseteq A^\omega$ and $d \geq 1$, define

$$L^{(d)} = \{\langle u_0 \rangle \dots \langle u_k \rangle \dots \mid u_0 \dots u_k \dots \in L, \forall i \geq 0 (u_i \in A^d)\}.$$

Lemma 4.7. *Let $L \subseteq A^\omega$ be regular. Define $\phi : (A^{(d)})^* \rightarrow M(L)^{(d)}$ by $\langle x \rangle \phi = x \eta_L$ for $\langle x \rangle \in (A^{(d)})^*$. Then ϕ recognizes $L^{(d)}$.*

Proof. We define morphism $\theta : \text{Im}(\phi) \rightarrow M(L^{(d)})$ such that $\phi\theta = \eta_{L^{(d)}}$, and thus according to Proposition 2.4, ϕ recognizes $L^{(d)}$.

Define θ by: for $m \in \text{Im}(\phi)$, $m\theta = \langle w \rangle \eta_{L^{(d)}}$, where $\langle w \rangle \in (A^{(d)})^*$ and $\langle w \rangle \phi = m$.

At first, we prove that θ is well defined. Let $\langle w_1 \rangle \phi = \langle w_2 \rangle \phi = m$, i.e. $w_1 \eta_L = w_2 \eta_L = m$. Then for all $x, y, z \in A^*$, $(xw_1yz)^\omega \in L$ iff $xw_2yz^\omega \in L$ and $(x(yw_1z)^\omega)^\omega \in L$ iff $x(yw_2z)^\omega \in L$, thus for all $\langle x \rangle, \langle y \rangle, \langle z \rangle \in (A^{(d)})^*$, $(\langle x \rangle \langle w_1 \rangle \langle y \rangle \langle z \rangle)^\omega \in L^{(d)}$ iff $\langle x \rangle \langle w_2 \rangle \langle y \rangle \langle z \rangle^\omega \in L^{(d)}$ and $(\langle x \rangle (\langle y \rangle \langle w_1 \rangle \langle z \rangle)^\omega)^\omega \in L^{(d)}$ iff $\langle x \rangle (\langle y \rangle \langle w_2 \rangle \langle z \rangle)^\omega \in L^{(d)}$, i.e. $\langle w_1 \rangle \approx_{L^{(d)}} \langle w_2 \rangle$, $\langle w_1 \rangle \eta_{L^{(d)}} = \langle w_2 \rangle \eta_{L^{(d)}}$, so θ is well defined.

Evidently for all $\langle w \rangle \in (A^{(d)})^*$, $\langle w \rangle \phi \theta = \langle w \rangle \eta_{L^{(d)}}$. \square

Lemma 4.8. Suppose that $L = \bigcup_{i=1}^m X_i(Y_i)^\omega$, where $X_i, Y_i \in \text{QSF}^F$, $Y_i \subseteq A^+$ and $Y_i Y_i \subseteq Y_i$. Then there is $d \geq 1$ such that all those X_i and Y_i can be constructed from the singleton languages $\{a\} (a \in A)$ and the language $(A^d)^*$.

Proof. Since $X_i, Y_i \in \text{QSF}^F$, then there are d_{X_i} and d_{Y_i} such that X_i and Y_i are constructed from the singleton languages $\{a\}$ and the language $(A^{d_{X_i}})^*$.

Let $d = \text{lcm}\{d_{X_i}, d_{Y_i} \mid 1 \leq i \leq m\}$. Then similar to the proof of Proposition 4.2, we can prove that X_i and Y_i can be constructed from singleton languages $\{a\}$ and the language $(A^d)^*$. \square

Lemma 4.9. Suppose that $L \subseteq (A^{(d)})^*$ is star free for some $d \geq 1$, then $L' = \{x \mid x \in (A^d)^*, \langle x \rangle \in L\}$ is quasi-star-free.

Proof. Since $L \subseteq (A^{(d)})^*$ is star free, it can be constructed from singleton languages $\{\langle u \rangle\} (u \in A^d)$ and the language $(A^{(d)})^*$ by union, complementation and concatenation.

By replacing $\{\langle u \rangle\} (u = a_0 \dots a_{d-1})$ by $\{a_0\} \dots \{a_{d-1}\}$; $(A^{(d)})^*$ by $(A^d)^*$; $L_1 \cup L_2$ by $L'_1 \cup L'_2$; $(A^{(d)})^* - L_1$ by $(A^d)^* - L'_1$ (namely $A^* - ((A^* - (A^d)^*) \cup L'_1)$); and $L_1 L_2$ by $L'_1 L'_2$ during the construction procedure of L , we can get the construction procedure of L' (where $L_1, L_2 \subseteq (A^{(d)})^*$ and L'_1, L'_2 are the languages of $(A^d)^*$ corresponding to L_1 and L_2 respectively). Thus L' can be constructed from singleton languages $\{a\}$ and the language $(A^d)^*$ by union, complementation and concatenation. Consequently it is quasi-star-free by definition. \square

Lemma 4.10. Let $L \subseteq A^\omega$. Then L is definable in $\text{FO}[C]$ iff there is some $d \geq 1$ such that $L^{(d)}$ is definable in $\text{FO}[<]$.

Lemma 4.11. Let $L \subseteq A^\omega$. Then L is definable in $\text{LTL}[C]$ iff L is definable in $\text{FO}[C]$.

Remark 4.12. The proofs of Lemma 4.10 and Lemma 4.11 are totally similar to the proofs of the same results for finite words (Proposition 6.5, Proposition 6.7 and Theorem 7.5 in [5]). Consequently we omit the proofs of them here.

Now we prove Theorem 4.6.

Proof of Theorem 4.6. At first we prove the equivalence of (ii) and (iii). According to Lemma 4.11, (v) and (vi) are equivalent. Then if we have proved the equivalence

of (i),(ii),(iv) and (v), the proof would be completed. We prove the equivalence of (i),(ii), (iv) and (v) by proving the equivalence of (i),(ii),(v) and equivalence of (ii),(v) respectively.

(ii) \Rightarrow (iii):

Suppose that $L \subseteq A^\omega$ is quasi-aperiodic, i.e. $M(L)^{(d)}$ is aperiodic for some $d \geq 1$. Now we show that for all $t \geq 0$, $A^t \eta_L$ contains no nontrivial group.

To the contrary suppose that there is some $t \geq 0$ such that $A^t \eta_L$ contains a nontrivial group. Obviously $t \geq 1$. Select an element m of order $k > 1$ from the group, then $G = \{m, \dots, m^k\}$ is also a nontrivial group in $A^t \eta_L$. Hence there are $u, v \in A^t$ such that $u \eta_L = m$, $v \eta_L = m^k$.

Consider $A^{tkd} \eta_L \subseteq M(L)^{(d)}$. It is easy to see that $m^i = (v^{k(d-1)} (u^i v^{k-i})) \eta_L \in A^{tkd} \eta_L$, thus $G \subseteq A^{tkd} \eta_L \subseteq M(L)^{(d)}$, $M(L)^{(d)}$ contains a nontrivial group. Because a monoid is aperiodic iff it contains no nontrivial group, we have that $M(L)^{(d)}$ isn't aperiodic, a contradiction.

(iii) \Rightarrow (ii):

The main idea is from the proof of Theorem 3 in [2].

Suppose that $M(L)$ is finite and for all $t \geq 0$, $A^t \eta_L$ contains no nontrivial group.

For each nontrivial group G contained in $M(L)$ pick a nonempty word v_G such that $v_G \eta_L$ is the identity of G . Let d be a common multiple of the lengths of all these v_G . Now we show that $M(L)^{(d)}$ is aperiodic.

To the contrary suppose that $M(L)^{(d)}$ isn't aperiodic. Because a monoid is aperiodic iff it contains no nontrivial group, then there is a nontrivial group in $M(L)^{(d)}$. Select an element m of order $k > 1$ from the group, then $G = \{m, \dots, m^k\}$ is also a nontrivial group in $M(L)^{(d)}$. Select some $v \in (A^d)^*$ such that $v \eta_L = m$. From the selection of d , we know $|v|$ (the length of v) is a multiple of $|v_G|$, thus there is some power w of v_G such that $|v| = |w|$. Let $t = k|v|$, then $m^j = (v^j w^{k-j}) \eta_L \in A^t \eta_L$, so $G \subseteq A^t \eta_L$, a contradiction.

Therefore we have proved the equivalence of (ii) and (iii).

Now we prove the equivalence of (i), (ii), (v).

(i) \Rightarrow (ii):

Suppose that L can be constructed from language A^ω by finite applications of operations of union, complementation, and concatenation on the left by quasi-star-free languages of A^* . Then according to Proposition 4.2, there is $d \geq 1$ such that quasi-star-free languages of A^* used in the construction of L can be constructed from singleton languages $\{a\}$ and the language $(A^d)^*$.

Now we prove that $M(L)^{(d)}$ is aperiodic by induction on the construction procedure of L .

Induction base: $L = A^\omega$, then $M(L) = \{e\}$, where e is the identity of $M(L)$. Obviously $M(L)^{(d)} = \{e\}$, then it is aperiodic.

Induction step:

Case $L = A^\omega - L_1$: From induction hypothesis, $M(L_1)^{(d)}$ is aperiodic. Since it is not hard to see that $M(L) = M(L_1)$ and $\eta_L = \eta_{L_1}$ from the definition of syntactic monoid and syntactic morphism of ω -languages, $M(L)^{(d)}$ is aperiodic as well.

Case $L = L_1 \cup L_2$: From induction hypothesis, $M(L_i)^{(d)}$ ($i = 1, 2$) are aperiodic, then according to Proposition 4.5, there are n_i ($i = 1, 2$) such that for all $n \geq n_i$ and $u, x, y, z \in A^*$ with $|u| \equiv 0 \pmod d$, $(xu^n yz^\omega \in L_i \text{ iff } xu^{n+1} yz^\omega \in L_i)$ and $(x(yu^n z)^\omega \in L_i \text{ iff } x(yu^{n+1} z)^\omega \in L_i)$.

Let $n_0 = \max\{n_1, n_2\}$. Now we show that for all $n \geq n_0$ and $u, x, y, z \in A^*$ with $|u| \equiv 0 \pmod d$, $(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L)$ and $(x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L)$. Then according to Proposition 4.5 we conclude that $M(L)^{(d)}$ is aperiodic.

Suppose that $xu^n yz^\omega \in L$, then $xu^n yz^\omega \in L_i$ for some $i = 1, 2$. Thus $xu^{n+1} yz^\omega \in L_i$ since $n \geq n_0 \geq n_i$, so $xu^{n+1} yz^\omega \in L$. The proof of $xu^{n+1} yz^\omega \in L$ implies $xu^n yz^\omega \in L$ is similar.

Suppose that $x(yu^n z)^\omega \in L$, then $x(yu^n z)^\omega \in L_i$ for some $i = 1, 2$. Thus $x(yu^{n+1} z)^\omega \in L_i$ since $n \geq n_0 \geq n_i$, so $x(yu^{n+1} z)^\omega \in L$. The proof of $x(yu^{n+1} z)^\omega \in L$ implies $x(yu^n z)^\omega \in L$ is similar.

Case $L = L_1 L_2$: where $L_1 \subseteq A^*$ and $L_2 \subseteq A^\omega$. According to Proposition 3.5, L_1 is quasi-aperiodic, then there is n_1 such that for all $n \geq n_1$, $xy^n z \in L_1$ iff $xy^{n+1} z \in L_1$ for all $x, y, z \in A^*$ with $|y| = 0 \pmod d$. From induction hypothesis, $M(L_2)^{(d)}$ is aperiodic, thus there is n_2 such that for all $n \geq n_2$, $u, x, y, z \in A^*$ with $|u| = 0 \pmod d$, $(xu^n yz^\omega \in L_2 \text{ iff } xu^{n+1} yz^\omega \in L_2)$ and $(x(yu^n z)^\omega \in L_2 \text{ iff } x(yu^{n+1} z)^\omega \in L_2)$.

Let $n_0 = n_1 + n_2 + 1$. It is sufficient to show that for all $n \geq n_0$ and $u, x, y, z \in A^*$ with $|u| = 0 \pmod d$, $(xu^n yz^\omega \in L \text{ iff } xu^{n+1} yz^\omega \in L)$ and $(x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L)$ in order to prove that $M(L)^{(d)}$ is aperiodic according to Proposition 4.5.

(a) Suppose that $n \geq n_0, u, x, y, z \in A^*$ with $|u| = 0 \pmod d$, and $xu^n yz^\omega \in L$. We show that $xu^{n+1} yz^\omega \in L$.

Since $xu^n yz^\omega \in L = L_1 L_2$, $xu^n yz^\omega$ has a decomposition vw such that $v \in L_1$ and $w \in L_2$. There are the following cases:

- $v = x_1, w = x_2 u^n yz^\omega$ with $x = x_1 x_2$;
- there are $h, k \geq 0, u_1, u_2 \in A^*$ such that $v = xu^h u_1, w = u_2 u^k yz^\omega$ with $n = h + k + 1, u = u_1 u_2$;
- $v = xu^n y_1, w = y_2 z^\omega$ with $y = y_1 y_2$;
- there are $p \geq 0, z_1, z_2 \in A^*$ such that $v = xu^n yz^p z_1, w = z_2 z^\omega$ with $z = z_1 z_2$.

Here we take the second case as an example, the discussions of the other cases are similar. In the second case, because $h + k + 1 \geq n_1 + n_2 + 1$, then $h \geq n_1$ or $k \geq n_2$, thus $xu^{h+1} u_1 \in L_1$ or $u_2 u^{k+1} yz^\omega \in L_2$, then $xu^{n+1} yz^\omega \in L_1 L_2 = L$.

The proof of $xu^{n+1} yz^\omega \in L$ implies $xu^n yz^\omega \in L$ is similar to (a).

(b) Suppose that $n \geq n_0, u, x, y, z \in A^*$ with $|u| = 0 \pmod d$, and $x(yu^n z)^\omega \in L$. We show that $x(yu^{n+1} z)^\omega \in L$.

Since $x(yu^n z)^\omega \in L = L_1 L_2$, $x(yu^n z)^\omega$ has a decomposition vw such that $v \in L_1$ and $w \in L_2$. There are the following cases:

- $v = x_1, w = x_2 (yu^n z)^\omega$ with $x = x_1 x_2$;

- there are $p \geq 0, y_1, y_2 \in A^*$ such that $v = x(yu^n z)^p y_1, w = (y_2 u^n z)(yu^n z)^\omega$ and $y = y_1 y_2$;
- there are $p, h, k \geq 0, u_1, u_2 \in A^*$ such that $v = x(yu^n z)^p (yu^h u_1), w = (u_2 u^k z)(yu^n z)^\omega$ with $n = h + k + 1, u = u_1 u_2$;
- there are $p \geq 0, z_1, z_2 \in A^*$ such that $v = x(yu^n z)^p (yu^n z_1), w = z_2 (yu^n z)^\omega, z = z_1 z_2$;

Here we take the third case as an example, the discussions of the other cases are similar.

Since $n \geq n_0 = n_1 + n_2 + 1 \geq n_i (i = 1, 2)$, then $x(yu^{n+1} z)^p (yu^h u_1) \in L_1$ and $(u_2 u^k z)(yu^{n+1} z)^\omega \in L_2$. Because $h + k + 1 \geq n_1 + n_2 + 1$, we have $h \geq n_1$ or $k \geq n_2$. Thus $x(yu^{n+1} z)^p (yu^{h+1} u_1) \in L_1$ or $(u_2 u^{k+1} z)(yu^{n+1} z)^\omega \in L_2$. Consequently

$$x(yu^{n+1} z)^p (yu^{h+1} u_1) (u_2 u^k z)(yu^{n+1} z)^\omega \in L_1 L_2$$

or

$$x(yu^{n+1} z)^p (yu^h u_1) (u_2 u^{k+1} z)(yu^{n+1} z)^\omega \in L_1 L_2.$$

Namely, $x(yu^{n+1} z)^\omega \in L_1 L_2 = L$.

The proof of $x(yu^{n+1} z)^\omega \in L$ implies $x(yu^n z)^\omega \in L$ is similar to (b).

(ii) \Rightarrow (v):

Suppose L is quasi-aperiodic, then there is $d \geq 1$ such that $M(L)^{(d)}$ is aperiodic, then according to Lemma 4.7, $L^{(d)}$ is aperiodic, thus L is definable in $\text{FO}[C]$ according to Lemma 4.10.

(v) \Rightarrow (i):

Suppose $L \subseteq A^\omega$ is definable in $\text{FO}[C]$, then according to Lemma 4.10, there is $d \geq 1$ such that $L^{(d)} \subseteq (A^{(d)})^\omega$ can be expressed in $\text{FO}[<]$. According to Proposition 2.6, $L^{(d)}$ is star-free, i.e. it can be constructed from $(A^{(d)})^\omega$ by union, complementation and concatenation on the left by star free languages of $(A^{(d)})^*$.

By replacing $L_1 \cup L_2, (A^{(d)})^\omega - L_1$, and $L_1 L_2$ by $L'_1 \cup L'_2, (A^{(d)})^\omega - L'_1$ and $L'_1 L'_2$ respectively during the construction of $L^{(d)}$ (where L'_1, L'_2 are languages of $(A^{(d)})^*$ or $(A^{(d)})^\omega$ corresponding to L_1 and L_2 respectively), we can get the construction procedure for L . Moreover, according to Lemma 4.9, languages of $(A^{(d)})^*$ used in the left concatenation during the construction of L must be quasi-star-free. Then we can conclude that L can be constructed from A^ω (namely $(A^{(d)})^\omega$) by union, complementation and concatenation on the left by quasi-star-free languages of A^* , i.e., L is quasi-star-free.

Therefore we have proved the equivalence of (i),(ii),(v).

Now we prove the equivalence of (ii),(iv) and complete the proof of the theorem.

(ii) \Rightarrow (iv):

Suppose L is quasi-aperiodic, i.e. there is $d \geq 1$ such that $M(L)^{(d)}$ is aperiodic.

According to Lemma 4.7, $L^{(d)}$ is aperiodic. Thus by Proposition 2.6, $L^{(d)} =$

$$\bigcup_{i=1}^m X_i Y_i^\omega, \text{ where } X_i \subseteq (A^{(d)})^*, Y_i \subseteq (A^{(d)})^+ \text{ are star free, and } Y_i Y_i \subseteq Y_i.$$

Let $X'_i = \{x \mid x \in (A^d)^*, \langle x \rangle \in X_i\}$, $Y'_i = \{y \mid y \in (A^d)^*, \langle y \rangle \in Y_i\}$, then $L = \bigcup_{i=1}^m X'_i (Y'_i)^\omega$. Evidently $Y'_i Y'_i \subseteq Y'_i$. Since $X_i, Y_i \subseteq (A^{(d)})^*$ are star free, then according to Lemma 4.9, X'_i and Y'_i are quasi-star-free.

(iv) \Rightarrow (ii):

Suppose that $L = \bigcup_{i=1}^m X_i (Y_i)^\omega$, where $X_i \subseteq A^*, Y_i \subseteq A^+$ are quasi-star-free languages, and $Y_i Y_i \subseteq Y_i$. Then according to Lemma 4.8, there is $d \geq 1$ such that X_i, Y_i can be constructed from singleton languages $\{a\} (a \in A)$ and the language $(A^d)^*$.

Because X_i is quasi-star-free, according to Proposition 3.5, X_i is quasi-aperiodic, i.e. there is $n_0 \in N$ such that for all $n \geq n_0$ and $x, y, z \in A^*$ with $|y| \equiv 0 \pmod d$, $xy^n z \in X_i$ iff $xy^{n+1} z \in X_i$. Denote this n_0 as $n_0(X_i)$. Similarly we have $n_0(Y_i)$ for Y_i . Moreover, since X_i, Y_i are quasi-star-free, $X_i Y_i$ is quasi-star-free as well, and we let $n_0(X_i Y_i) \geq n_0(X_i) + n_0(Y_i) + 1$ for $X_i Y_i$ such that for all $n \geq n_0(X_i Y_i)$ and $x, y, z \in A^*$ with $|y| \equiv 0 \pmod d$, $xy^n z \in X_i Y_i$ iff $xy^{n+1} z \in X_i Y_i$.

Let $N_0 = 1 + 2 \max\{n_0(X_i Y_i) \mid 1 \leq i \leq m\}$. It is sufficient to show that for all $n \geq N_0$ and $u, x, y, z \in A^*$ with $|u| \equiv 0 \pmod d$, $(xu^n y z^\omega \in L \text{ iff } xu^{n+1} y z^\omega \in L)$ and $(x(yu^n z)^\omega \in L \text{ iff } x(yu^{n+1} z)^\omega \in L)$ in order to prove that L is quasi-aperiodic (according to Proposition 4.5).

(a) Suppose that $n \geq N_0$, $u, x, y, z \in A^*$, $|u| \equiv 0 \pmod d$, and $xu^n y z^\omega \in L$, we show that $xu^{n+1} y z^\omega \in L$.

Because $L = \bigcup_{i=1}^m X_i (Y_i)^\omega$, $xu^n y z^\omega \in X_i (Y_i)^\omega$ for some i . Then there is $p, p', q, q' \geq 0$, $z_1, z_2 \in A^*$ such that $z = z_1 z_2$, $xu^n y z^{p'} z_1 \in X_i Y_i^p$ and $z_2 z^{q'} z_1 \in Y_i^q$. If $p = 0$, then $xu^{n+1} y z^{p'} z_1 \in X_i$ since $n \geq N_0 \geq n_0(X_i Y_i) \geq n_0(X_i)$, $xu^{n+1} y z^\omega = (xu^{n+1} y z^{p'} z_1) (z_2 z^{q'} z_1)^\omega \in X_i ((Y_i)^q)^\omega = X_i Y_i^\omega \subseteq L$. In the case of $p > 0$, $X_i Y_i^p \subseteq X_i Y_i$ follows from that assumption $Y_i Y_i \subseteq Y_i$, so $xu^{n+1} y z^{p'} z_1 \in X_i Y_i$ since $n \geq N_0 \geq n_0(X_i Y_i)$; then $xu^{n+1} y z^\omega = (xu^{n+1} y z^{p'} z_1) (z_2 z^{q'} z_1)^\omega \in X_i Y_i ((Y_i)^q)^\omega = X_i (Y_i)^\omega \subseteq L$.

The proof of $xu^{n+1} y z^\omega \in L$ implies $xu^n y z^\omega \in L$ is similar to (a).

(b) Suppose that $n \geq N_0$, $u, x, y, z \in A^*$, $|u| \equiv 0 \pmod d$, and $x(yu^n z)^\omega \in L$, we show that $x(yu^{n+1} z)^\omega \in L$.

Because $L = \bigcup_{i=1}^m X_i (Y_i)^\omega$, $x(yu^n z)^\omega \in X_i Y_i^\omega$ for some i . Then there are $p, p', q, q' \geq 0$, $v_1, v_2 \in A^*$ such that $x(yu^n z)^{p'} v_1 \in X_i Y_i^p$, $v_2 (yu^n z)^{q'} v_1 \in Y_i^q$, $v_1 v_2 = yu^n z$.

Here we prove for the case of $p > 0$, the case of $p = 0$ can be proved similarly.

Suppose that $p > 0$.

Since $Y_i Y_i \subseteq Y_i$, we have $X_i Y_i^p \subseteq X_i Y_i$, $Y_i^q \subseteq Y_i$.

Because $n \geq N_0 \geq n_0(X_i, Y_i) \geq n_0(Y_i)$, we have that $x(yu^{n+1} z)^{p'} v_1 \in X_i Y_i$ and $v_2 (yu^{n+1} z)^{q'} v_1 \in Y_i$.

Now we discuss the following three cases of v_1 and v_2 .

- $v_1 = y_1, v_2 = y_2 u^n z, y = y_1 y_2$;
- $v_1 = y u^n z_1, v_2 = z_2, z = z_1 z_2$;
- $v_1 = y u^h u_1, v_2 = u_2 u^k z$, with $h + k + 1 = n$ and $u = u_1 u_2$.

Here we take the third case as the example, the discussions of other cases are similar.

Case $v_1 = y u^h u_1, v_2 = u_2 u^k z$, with $h + k + 1 = n$ and $u = u_1 u_2$:

Since $n \geq N_0 \geq 1 + 2n_0(X_i Y_i)$, we have $h \geq n_0(X_i Y_i)$ or $k \geq n_0(X_i Y_i)$.

If $h \geq n_0(X_i Y_i)$, then

$$x(yu^{n+1}z)^{p'}(yu^{h+1}u_1) \in X_i Y_i, (u_2 u^k z)(yu^{n+1}z)^{q'}(yu^{h+1}u_1) \in Y_i.$$

Thus

$$x(yu^{n+1}z)^\omega = \left(x(yu^{n+1}z)^{p'}(yu^{h+1}u_1) \right) \left((u_2 u^k z)(yu^{n+1}z)^{q'}(yu^{h+1}u_1) \right)^\omega \in X_i Y_i^\omega.$$

If $k \geq n_0(X_i Y_i)$, then $(u_2 u^{k+1} z)(yu^{n+1}z)^{q'}(yu^h u_1) \in Y_i$. Thus

$$x(yu^{n+1}z)^\omega = \left(x(yu^{n+1}z)^{p'}(yu^h u_1) \right) \left((u_2 u^{k+1} z)(yu^{n+1}z)^{q'}(yu^h u_1) \right)^\omega \in X_i Y_i^\omega.$$

The proof of $x(yu^{n+1}z)^\omega \in L$ implies $x(yu^n z)^\omega \in L$ is similar to (b). \square

5 Conclusions and Remarks

In this paper quasi-star-free languages on infinite words (QSF^I) are defined and studied. Quasi-star-free languages on finite words (QSF^F) have been studied in [2, 5], and our work in this paper is an extension of those results for QSF^F in [2, 5].

The extension of results of QSF^F to QSF^I should be more useful for the characterizations of the expressive power of temporal logics since temporal logics are usually interpreted on infinite words in order to describe temporal properties of concurrent systems. One of the examples is the characterizations of expressive power of fragments of linear μ -calculus [8] (known as νTL). The “next” operators within the scope of the fixed points of νTL formulas act like the FO[C] predicates “ $C_d^r(x)$ ” and LTL[C] operators “ $U^{(d,r)}$ ”, e.g. νTL formula $\nu Q.p_a \wedge XXXQ$ defines language $(\{a\}A)^\omega$, which can be defined by FO[C] sentence $\forall x(C_2^0(x) \rightarrow p_a(x))$ and LTL[C] formula $\neg(Tru U^{(2,0)} \neg p_a)$ respectively, as we have noticed in Section 4.2.

Acknowledgements. I would like to thank anonymous referees for their comments and suggestions. Moreover, I would like to thank Prof. Wenhui Zhang for his reviews on this paper and discussions with me.

References

- [1] A. Arnold. A syntactic congruence for rational ω -languages. *Theoretical Computer Science* 39, 333-335, 1985.
- [2] D.A. Barrington, K. Compton, H. Straubing, D. Thérien. Regular languages in NC^1 . *Journal of Computer and System Sciences* 44, 478-499, 1992.
- [3] J. R. Büchi, On a decision method in restricted second order arithmetic. In: E. Nagel et al., eds., *Proc Internat. Congr. on Logic, Methodology and Philosophy of Science* (Stanford Univ. Press, Stanford, CA), 1-11, 1962.
- [4] J.Cohen, D.Perrin, J.E.Pin. On the expressive power of temporal logic for finite words, *Journal of Computer and System Sciences* 46, 271-294, 1993.
- [5] Z. Ésik, M. Ito. Temporal logic with cyclic counting and the degree of aperiodicity of finite automata. *Acta Cybernetica* 16, 1-28, 2003.
- [6] S. Eilenberg. *Automata, Languages and Machines*, Volume B. Academic Press, 1976.
- [7] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the Temporal Analysis of Fairness. In *Conference Record of the 7th ACM Symposium on Principles of Programming Languages (POPL'80)*, 163-173, 1980.
- [8] R. Kaivola. Axiomatising linear time μ -calculus. In *6th International Conference of Concurrency theory, LNCS 962*: 423-437, 1995.
- [9] H. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, Los Angeles, California, USA, 1968.
- [10] G.. Lallement. *Semigroups and combinatorial applications*. Wiley, New York, 1979.
- [11] R. McNaughton and S. Papert. *Counter-free automata*. MIT Press, 1971.
- [12] D. Perrin. Finite automata. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, 1-53, Elsevier Science Publishers, Amsterdam, 1990.
- [13] D. Perrin. Recent results on automata and infinite words. In *11th MFCS, LNCS 176*, 134-148, 1984.
- [14] J.E.Pin. *Varieties of formal languages*. North Oxford Academic, London, Plenum, New York, 1986.
- [15] A.Pnueli. The temporal logic of programs. *Proc. 18th FOCS*, Providence, RI, 46-57, 1977.
- [16] D.Perrin, J.E.Pin. *Infinite Words*. Pure and Applied Mathematics, Vol 141, Elsevier, 2004, ISBN 0-12-532111-2.

- [17] M.P Schutzenberger. On finite monoids having only trivial subgroups. *Information and Computation* 8, 190-194, 1965.
- [18] W. Thomas. Star-free regular sets of ω -sequences. *Inform. and Control* 42, 148-156, 1979.
- [19] W. Thomas. A combinatorial approach to the theory of ω -automata. *Inform. and Control* 48, 261-283, 1981.
- [20] W. Thomas. Computation tree logic and regular ω -languages. In *REX Workshop 1988: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354, 690-713, 1989.
- [21] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, 133-192. Elsevier Science Publishers, Amsterdam, 1990.
- [22] H. Straubing. *Finite Automata, Formal Logic and Circuit Complexity*. Birkhauser, 1994.
- [23] P. Wolper. Temporal logic can be more expressive. *Inform. and Control* 56, 72-99, 1983.

Received September, 2004

The lexicographic decision function

József Dombi* and Nándor Vincze†

Abstract

In this paper the lexicographic decision process is presented in a unified way. We construct a lexicographic decision function using a universal preference function and a unary function. This construction incorporates the different outranking approaches, the lexicographic decision process and the utility based decision making models. Finally we consider the connection of the lexicographic decision method and the Arrow paradox.

1 Introduction

In this section we describe the concept of the lexicographic method. We use the terminology of P.C. Fishburn, see [6]. The general concept of a finite lexicographic order involves a set $I = \{1, 2, \dots, n\}$ and an order relation \prec_i on a nonempty set X_i for each $i \in I$. We let \sim_i denote the symmetric complement of \prec_i so that $x_i \sim_i y_i$ if and only if $(x_i \prec_i y_i \text{ or } y_i \prec_i x_i)$ does not hold. With $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$, y precedes x lexicographically under the natural order $<$ on I and with respect to the \prec_i or $x <^L y$ for short, iff $\{i : i \in I \text{ and } (x_i \prec_i y_i \text{ or } y_i \prec_i x_i)\}$ is nonempty, and $x_i \prec_i y_i$ for the first (smallest) i in this set.

For this reason a lexicographic order $<^L$ is also referred to as an order by first difference.

An example of a lexicographic order arises from the alphabetical order of words in a dictionary or lexicon. To show this let $I = \{1, 2, \dots, n\}$, let $X_i = A = \{\emptyset, a, b, \dots, z\}$ with $\emptyset \prec_i a \prec_i b \prec_i \dots \prec_i z$ for each i , take n as large as the longest listed word, and let the English word $\alpha_1 \alpha_2 \dots \alpha_m$ with $m \leq n$ correspond to $(\alpha_1, \alpha_2, \dots, \alpha_m, \emptyset, \dots, \emptyset)$ in A^n . Then $<^L$ on the subset of A^n which corresponds to the "legitimate" words orders these words in their natural alphabetical order. For example, "as" precedes "ask" since $(a, s, \emptyset, \dots, \emptyset) <^L (a, s, k, \emptyset, \dots, \emptyset)$ which is to say that $a \sim_1 a$, $s \sim_2 s$, $\emptyset \prec_3 k$.

In multicriteria decision making the idea of the lexicographic decision consists of a hierarchy or ordered set of attributes or criteria. Decision alternatives are examined initially on the basis of the first or most important criterion. If more than one alternative is "best" or "satisfactory" on this basis, then these are compared under

*Department of Informatics, University of Szeged H-6720 Szeged, Árpád tér 2. Hungary

†Department of Engineering and Informatics, Szeged College Faculty of Food Engineering, University of Szeged H-6725 Szeged, Mars tér 20. Hungary

the second most important criterion and so forth. The principle of order by first difference says that one alternative is "better" than another iff the first is "better" than the second on the most important criterion on which they differ.

So let x and y be two alternatives (actions) and c_1, c_2, \dots, c_n be different criteria, x_i and y_i are the utilities (evaluations) of x and y . We identify x and y with their evaluation vector $x = (x_1, x_2, \dots, x_n)$, $y = (y_1, y_2, \dots, y_n)$. Then \prec_i is the order relation according to c_i on the set of alternatives.

We say that $x_i \sim_i y_i$ iff the alternatives x and y are indifferent according to the c_i , and we say that $x <^L y$ iff $x_i \sim_i y_i$ for $i \in \{1, 2, \dots, k-1\}$ where $0 \leq k-1 \leq n-1$ and $x_k \prec_k y_k$ in other words the alternative y is preferred to the alternative x , according to the criteria c_k .

The lexicographic decision method is a well adaptable method. It can arrange data of arbitrary scales, and it is suitable to evaluate a set of considerable alternatives. This method does not require the weight of criteria and in spite of its simplicity always arranges the alternatives, Rapcsák [18]. Some decision procedures have lexicographic decision rules to prevent ties, Temesi [20]. Sequential screening procedures illustrate another common application of the lexicographic idea. Candidates or alternatives are first screened under a given criterion (perhaps with the use of a test or an interview) and separated into "rejects" and "others". In terms of \prec_1 of the set of candidates, $x \sim y$ whenever both x and y are "rejects" or "others", with $x \prec_1 y$ when x is a "reject" and y is an "other". The "others" are then screened further by the second criterion or test and sorted into two groups. Of course the "rejects" from the first stage may not be tested for the second stage, but that is of no importance from the viewpoint of the lexicographic rule except from the standpoint of efficiency that it may promote. This process may continue through several more stages, perhaps including a ranking of all candidates who survive to the last stage. Another aspect of the using of lexicographic decision method is to avoid the intransitivity of preference. If \prec_i is a weak order for every i then $<^L$ is a weak order, if \prec_i is a linear order for every i then $<^L$ is also a linear order, but when \prec_i is a partial order for every i it does not follow that $<^L$ is a partial order, even if $<^L$ includes cycles. If $\prec_i \neq \emptyset$ then the lexicographic aggregation preserves transitivity, Fishburn [6], Solymosi [19]. About a general concept of the preference cycles and its representation, see Dombi, Vincze [4].

In the evaluation of alternatives, according to the c_k criteria the values x_k , and y_k would be numerical values or categories. In the case of categories the lexicographic decision can be characterized with weighted criteria. We will prove that there exists a weighted representation of lexicographic decision method on the real numbers. This yields a universal form: PROMETHEE, ELECTRE and utility are special cases of it, see Dombi [3].

It is important to note that the solution of many MCDM problems requires the application of two or three decision methods. For example when the groups of criteria needs different aggregation procedures. In our model we can give different decision making methods by changing the parameters.

We construct a weighted method to get the decision function of the lexicographic decision method. We choose the weights in such a way, that a range of alternatives

by c_k criteria could not be changed by $c_{k+1}, c_{k+2}, \dots, c_n$ criteria.

Finally we compare the conditions of the lexicographic decision method and the Arrow impossibility theorem.

The main aspect of our motivation is that the mentioned non-compensatory property arrange the criteria by their importancy and hence is the dictator in this decision model. So the dictatorship is an essential precept in this method.

In our paper we suppose that among the alternatives there are no two lexicographically equal.

2 The construction of the lexicographic decision function.

The lexicographic decision method is a seldom occurring theme in publications. For its numerical representation we could not find solution. It may follow from the negative results in this logic, for example the lexicographic order of the plane:

Theorem 1. *There does not exist any continuous $f(x, y)$ function, such that:*

$$(x, y) <^L (v, z) \text{ iff } f(x, y) < f(v, z).$$

Proof. Let the values x, x_1, x_2, y_1, y_2 be such that $x_1 < x < x_2$ and $y_1 < y_2$. We suppose that there exists continuous $f(x, y)$ function, for which:

$$(x, y) <^L (v, z) \text{ iff } f(x, y) < f(v, z).$$

Then for the mentioned values it is true, that:

$$(x, y_2) <^L (x_2, y_1) <^L (x_2, y_2) \text{ iff } f(x, y_2) <^L f(x_2, y_1) <^L f(x_2, y_2).$$

Because $f(x, y)$ is continuous, it is continuous at the point (x_2, y_2) .

Let ε be an arbitrarily fixed positive value such that

$$\varepsilon < f(x_2, y_2) - f(x_2, y_1).$$

Then there exists a value δ , such that:

$$\text{if } |(x_2, y_2) - (x, y_2)| < \delta \text{ then } f(x_2, y_2) - f(x, y_2) < \varepsilon.$$

but

$$f(x_2, y_2) - f(x, y_2) > f(x_2, y_2) - f(x_2, y_1) > \varepsilon$$

which contradicts that $f(x, y)$ is continuous. □

2.1 The preference and the modifier functions

In the introduction we shown the lexicographical decision concept. In this section we construct a lexicographical decision function. For the construction we use a general preference function $p(x, y)$ and a $\tau(x)$ modifier, (or threshold) function, which are the following, according to Dombi [3]:

$$p(x, y) = (y - x + 1)/2$$

$$\tau(x) = \begin{cases} 0 & \text{if } 0 \leq x < 1/2, \\ 1/2 & \text{if } x = 1/2, \\ 1 & \text{if } 1/2 < x \leq 1. \end{cases}$$

Let $A = \{a_1, a_2, \dots, a_m\}$ be the set of alternatives. Let $C = \{c_1, c_2, \dots, c_n\}$ be the set of the criteria, ordered by importancy. Let x_{ij} denote the evaluation (utility) of c_j criteria in the case of choosing a_i as an alternative, $0 \leq x_{ij} \leq 1$. The decision situation can be described with the following decision matrix:

	c_1	c_2	\dots	c_n
a_1	x_{11}	x_{12}	\dots	x_{1n}
a_2	x_{21}	x_{22}	\dots	x_{2n}
\vdots	\vdots	\vdots	\vdots	\vdots
a_m	x_{m1}	x_{m2}	\dots	x_{mn}

2.1.1 Properties of the preference function

Let $p(x, y) = P(y - x)$ consider as the function of $y - x$, and let $0 \leq x, y \leq 1$. Then $y - x \in [-1, 1]$. We get that:

$$\text{sign}(y - x) = \begin{cases} -1 & \text{if } 0 \leq p(x, y) < 1/2 \\ 0 & \text{if } p(x, y) = 1/2 \\ 1 & \text{if } 1/2 < p(x, y) \leq 1 \end{cases}$$

Then

$$P(\text{sign}(y - x)) = \begin{cases} 0 & \text{if } 0 \leq p(x, y) < 1/2 \\ 1/2 & \text{if } p(x, y) = 1/2 \\ 1 & \text{if } 1/2 < p(x, y) \leq 1 \end{cases}$$

As described in the introduction we identify the alternatives with its evaluation n-tuples, so we let

$$a_i = (x_{i1}, x_{i2}, \dots, x_{in}) \quad \text{and} \quad a_j = (x_{j1}, x_{j2}, \dots, x_{jn}).$$

To order the alternatives a_i and a_j with respect to criteria c_k , we set $x = x_{ik}$ and $y = x_{jk}$ in the preference function $p(x, y)$.

2.1.2 The composition of the preference and the modifier function.

Definition 1. We can define for every (a_i, a_j) pair the $p^*(a_i, a_j)$ preference n -tuple in the following manner. Let

$$p^*(a_i, a_j) = (\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) \quad \text{for} \quad \varepsilon_{ij}^k = \tau(p(x_{ik}, x_{jk})).$$

Then

$$\tau(p(x_{ik}, x_{jk})) = \begin{cases} 0 & \text{if } x_{ik} > x_{jk} \\ 1/2 & \text{if } x_{ik} = x_{jk} \\ 1 & \text{if } x_{ik} < x_{jk} \end{cases}$$

The indicators ε_{ij}^k can be considered as the elements of a pairwise comparison matrix with respect to the c_k criterion.

c_k	a_1	a_2	\dots	a_m
a_1	ε_{11}^k	ε_{12}^k	\dots	ε_{1m}^k
a_2	ε_{21}^k	ε_{22}^k	\dots	ε_{2m}^k
\vdots	\vdots	\vdots	\vdots	\vdots
a_m	ε_{m1}^k	ε_{m2}^k	\dots	ε_{mm}^k

All the elements in the main diagonal equal to 0.5. As mentioned before, we suppose, that among the alternatives there are no two lexicographically equal, so for each pair (a_i, a_j) $a_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $a_j = (x_{j1}, x_{j2}, \dots, x_{jn})$, there exist k_1 and k_2 such that $x_{ik_1} < x_{jk_1}$ and $x_{jk_2} < x_{ik_2}$.

2.2 The lexicographic decision function

The main result of this paper is the following Theorem:

Theorem 2. Let $A = \{a_1, a_2, \dots, a_m\}$ be the set of alternatives. Let $C = \{c_1, c_2, \dots, c_n\}$ be the set of criteria, ordered by importancy. Let x_{ij} denote the evaluation (utility) of criterion c_j in the case of choosing a_i as an alternative, $0 \leq x_{ij} \leq 1$. The decision situation can be described with the decision matrix:

	c_1	c_2	\dots	c_n
a_1	x_{11}	x_{12}	\dots	x_{1n}
a_2	x_{21}	x_{22}	\dots	x_{2n}
\vdots	\vdots	\vdots	\vdots	\vdots
a_m	x_{m1}	x_{m2}	\dots	x_{mn}

Let $p(x, y)$ be the preference function and $\tau(x)$ be the modifier, (or threshold) function as we defined in section 2.1.

Then there exists w_k weights $k = 1, 2, \dots, n$ such that the real numbers:

$$l_i = \frac{1}{m} \sum_{j=1}^m \tau \left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk})) \right), \quad i = 1, 2, \dots, m$$

satisfy that

$$l_i < l_j \quad \text{if and only if} \quad a_i >^L a_j.$$

So we construct the lexicographic decision function with the help of a weighting system. This function is non compensatory. This we give in the following. Next we give the weighting system.

Let the weight of c_i criterion be:

$$w_i = 1/2^i + 1/(n2^n)$$

It can be verified, that:

$$\sum_{k=1}^n w_k = 1.$$

The lexicographic decision function is constructed with the following function composition:

$$\tau \left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk})) \right) = \begin{cases} 0 & \text{if } a_i >^L a_j \\ 1 & \text{if } a_i <^L a_j \end{cases}$$

Since $\varepsilon_{ij}^k = \tau(p(x_{ik}, x_{jk}))$, we denote

$$\varepsilon_{ij} = \tau \left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk})) \right).$$

The following matrix provides the pairwise comparison matrix with respect to the weighted system of criteria $(c_1, w_1; c_2, w_2; \dots; c_n, w_n)$

(C, w)	a_1	a_2	\dots	a_m
a_1	ε_{11}	ε_{12}	\dots	ε_{1m}
a_2	ε_{21}	ε_{22}	\dots	ε_{2m}
\vdots	\vdots	\vdots	\vdots	\vdots
a_m	ε_{m1}	ε_{m2}	\dots	ε_{mm}

All the elements in the main diagonal equal always to 0.5.

Normalizing the lexicographic decision function we get real l_i in the interval $[0, 1]$.

$$l_i = \frac{1}{m} \sum_{j=1}^m \tau \left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk})) \right), \quad i = 1, 2, \dots, m$$

so that:

$$l_i < l_j \quad \text{iff} \quad a_i >^L a_j.$$

This sequence of real numbers is constructed in such a way, that for alternative a_i we aggregate the preferences between a_i and a_j for $j = 1, 2, \dots, i-1, i+1, \dots, m$.

This is the main idea of the global preference construction of the PROMETHEE method.

To prove the correctness of the construction, first we prove the correctness of the weighting.

Lemma 1. Let $\varepsilon_{ij}^k = \tau(p(x_{ik}, x_{jk}))$ as we defined it in 2.1.2. Then the following statements are true:

$$(1) \min_{i,j} \sum_{k=1}^n w_k \varepsilon_{ij}^k = 1/2 + 1/(n2^n) \text{ if } a_i <^L a_j, \text{ and it is minimal if}$$

$$(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (1, 0, \dots, 0).$$

$$(2) \max_{i,j} \sum_{k=1}^n w_k \varepsilon_{ij}^k = 1/2 - 1/(n2^n) \text{ if } a_i >^L a_j, \text{ and it is maximal if}$$

$$(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (0, 1, \dots, 1).$$

Proof (of Lemma 1).

(1) If $a_i <^L a_j$ and $\varepsilon_{ij}^k = \tau(p(x_{ik}, x_{jk}))$ then a preference n-tuple

$$(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (1/2, 1/2, \dots, 1/2, 1, \varepsilon_{ij}^{t+2}, \dots, \varepsilon_{ij}^n) \text{ for } 0 \leq t < n$$

has minimal non-zero element, if $\varepsilon_{ij}^{t+2} = \varepsilon_{ij}^{t+3} = \dots = \varepsilon_{ij}^n = 0$.

In this case:

$$\sum_{k=1}^n w_k \varepsilon_{ij}^k = 1/2 + (t/2 + 1)[1/(n2^n)].$$

It is minimal if $t = 0$. Then $(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (1, 0, \dots, 0)$ and the minimum is equal to $1/2 + 1/(n2^n)$.

(2) If $a_i >^L a_j$ then a preference n-tuple

$$(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (1/2, 1/2, \dots, 1/2, 0, \varepsilon_{ij}^{t+2}, \dots, \varepsilon_{ij}^n) \text{ for } 0 \leq k < n$$

has minimal zero element, if $\varepsilon_{ij}^{t+2} = \varepsilon_{ij}^{t+3} = \dots = \varepsilon_{ij}^n = 1$.

Then

$$\sum_{k=1}^n w_k \varepsilon_{ij}^k = 1/2 - (t/2 + 1)[1/(n2^n)].$$

This is maximal, if $t = 0$ and the maximum is $1/2 - 1/(n2^n)$.

Then

$$(\varepsilon_{ij}^1, \varepsilon_{ij}^2, \dots, \varepsilon_{ij}^n) = (0, 1, \dots, 1).$$

□

Proof (of Theorem 2). By Lemma 1 we get for the weighted sum that:

$$\begin{aligned} 0 &\leq \sum_{k=1}^n w_k \varepsilon_{ij}^k < 1/2 \quad \text{if } a_i >^L a_j \\ 1/2 &< \sum_{k=1}^n w_k \varepsilon_{ij}^k \leq 1 \quad \text{if } a_i <^L a_j \end{aligned}$$

Applying the modifier (or threshold) function $\tau(x)$ for this weighted sum, we obtain:

$$\tau\left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk}))\right) = \begin{cases} 0 & \text{if } a_i >^L a_j, \\ 1 & \text{if } a_i <^L a_j. \end{cases}$$

So $\tau\left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk}))\right)$ gives the lexicographic preference ordering between alternatives. So with this construction we get a decision function. Then we get:

$$\sum_{j=1}^m \tau\left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk}))\right) = |\{a_j : a_i <^L a_j\}| + \frac{1}{2}.$$

To transform this number to the $[0,1]$ interval, we get the real values:

$$l_i = (1/m) \sum_{j=1}^m \tau\left(\sum_{k=1}^n w_k \tau(p(x_{ik}, x_{jk}))\right)$$

for which

$$l_i < l_j \quad \text{if and only if} \quad a_i >^L a_j.$$

□

2.3 The lexicographic decision method as the limit of decision methods

Using the mentioned $\tau(x)$ threshold function, we construct the lexicographic decision function. This form is the general form of decision functions (for examples of PROMETHEE, ELECTRE and utility). In this formulation the form of the general modifier (threshold) function is, see Dombi [3]:

$$\tau_{p_1 p_2}(x) = \begin{cases} 0 & \text{if } 0 \leq x < p_1, \\ (x - p_1)/(p_2 - p_1) & \text{if } p_1 \leq x \leq p_2, \\ 1 & \text{if } p_2 < x \leq 1. \end{cases}$$

This function is linear in the interval $[p_1, p_2]$. Taking the limit of this function we obtain:

$$\lim_{p_1 \rightarrow \frac{1}{2}, p_2 \rightarrow \frac{1}{2}} \tau_{p_1 p_2}(x) = \tau(x).$$

So we get the lexicographic decision method as the limits of decision methods. These methods may be compensatorical or non compensatorical.

3 The lexicographic decision method, and the Arrow paradox

As mentioned in the introduction, the concept of arranging the criteria according to their importance and the lexicographic decision method is dictatorial. Because of this there may be connections between the lexicographic rule, and Arrow's impossibility theorem. But conditions of Arrow's impossibility theorem are applied to the voting situation, and so the lexicographic decision situation should be applied to a voting situation.

Let the evaluation of alternatives with respect to criterion c_i be $x_{1i}, x_{2i}, \dots, x_{ni}$. Let their order be $x_{1i}^* < x_{2i}^* < \dots < x_{ni}^*$, and set $x_{ki}^* = \frac{k}{n}$, so we get simply an ordering on alternatives by c_i . To transform the voting situation to multicriteria decision situation we map the individuals to criteria. In this section the profile is a weak order on the alternatives based on a criteria (or individual). The social welfare function is a decision function which aggregates the criterion (or individual) ordering. Let R be the set of all possible weak orders on the set of alternatives. We say that an individual is a dictator if its preferences become automatically social preferences.

The axioms and conditions of the Arrow paradox are the following, see Hwang, Lin [13]:

Axiom 1 (The preference relation is strongly complete). For all a_i and a_j either a_i 'is preferred or indifferent to' a_j or a_j 'is preferred or indifferent to' a_i .

Axiom 2 (The preference relation is transitive). For all a_i and a_j and a_k : a_i 'is preferred or indifferent to' a_j and a_j 'is preferred or indifferent to' a_k imply a_i 'is preferred or indifferent to' a_k .

Condition 1 (Universal domain). The social welfare function (decision function) f is defined for all possible profiles of individual (criteria).

Condition 2 (The weak Pareto concept). If $a_k, a_l \in A$ and $a_k \prec_i a_l$ for $i = 1, 2, \dots, n$ then $a_k \prec^L a_l$.

Condition 3 (Independence from irrelevant alternatives). $R^{(a_i, a_j)} = F^{(a_i, a_j)}(p^{(a_i, a_j)})$, for every pair $(a_i, a_j) \in A \times A$, where $R^{(a_i, a_j)}$, $F^{(a_i, a_j)}$, $p^{(a_i, a_j)}$ are the contraction of the social preference ordering, the social welfare function (i.e. the social decision function), and the p profile, to the pair (a_i, a_j) .

Condition 4 (Non-dictatorship). There is no dictator in the society, i.e. there is no individual that whenever he prefers a_i to a_j for any a_i and a_j society does likewise regardless of the preferences of other individuals.

Theorem 3 (General possibility theorem (Arrow)). If there are at least two individuals, and three alternatives, which the members of the society are free to order in any way, (condition 1.) then every social welfare function satisfying condition 2 and 3 and yielding a social ordering satisfying axioms I. and II. must be dictatorial.

It means that if a given social welfare function satisfies conditions 1-4, then a contradiction arises.

It can be seen that the lexicographic decision function satisfies Axioms 1-2 and Conditions 1-3.

We now consider the formulation in which there are preference orders \prec_i on the set of alternatives for each criteria along with holistic order $<^L$ on A , see Fishburn [5],[6], May [15], Plott [17].

We shall refer to an $n + 1$ tuple $(\prec_1, \prec_2, \dots, \prec_n, <^L)$ of weak orders on A as a situation. Then we consider the possibility that any one of a number of potential situation might arise.

Theorem 4. *Let us suppose that A contains at least three alternatives, (A is otherwise unlimited) and every n -tuple $(\prec_1, \prec_2, \dots, \prec_n)$ of weak orders on A appears in at least one situation. Then preferences are lexicographic, iff the following hold for all situations $(\prec_1, \prec_2, \dots, \prec_n, <^L)$ and $(\prec'_1, \prec'_2, \dots, \prec'_n, <'^L)$ and all $a_j, a_k \in A$:*
 $(a_j \sim_i a_k \text{ for all } i) \Rightarrow a_j \sim a_k$; $(a_j \succsim_i a_k \text{ for all } i) \& (a_j \prec_i a_k \text{ for some } i) \Rightarrow a_j <^L a_k$,
 and $(a_j \prec_i a_k \text{ iff } a_j \prec'_i a_k) \& (a_k \prec_i a_j \text{ iff } a_k \prec'_i a_j \text{ for all } i) \Rightarrow (a_j <^L a_k \text{ iff } a_k <'^L a_j) \& (a_k <^L a_j \text{ iff } a_j <'^L a_k)$.

Now we compare the axioms and conditions of the lexicographic decision method, and the Arrow paradox. We shall refer to the lexicographic method and the Arrow Paradox in this comparison by letters L and A, respectively.

1. Preference completeness and transitivity

L: The preference is a weak order, so it is strongly complete and transitive.

A: The preference is strongly complete, and transitive.

2. Universal domain

L: Every n -tuple $(\prec_1, \prec_2, \dots, \prec_n)$ of weak orders on A appears in at least one situation.

A: The social welfare function (decision function) f is defined for all possible profiles of individual (criteria).

3. The Pareto concepts

L: *The strong Pareto concept* $(a_j \sim_i a_k \text{ for all } i) \Rightarrow a_j \sim a_k$; $(a_j \succsim_i a_k \text{ for all } i, \& a_j \prec_i a_k \text{ for some } i) \Rightarrow a_j <^L a_k$,

A: *The weak Pareto concept* If $a_k, a_l \in A$ and $a_k \prec_i a_l$ for $i = 1, 2, \dots, n$ then $a_k <^L a_l$.

4. Independence from irrelevant alternatives

L: $(a_j \prec_i a_k \text{ iff } a_j \prec'_i a_k) \& (a_k \prec_i a_j \text{ iff } a_k \prec'_i a_j \text{ for all } i) \Rightarrow (a_j <^L a_k \text{ iff } a_k <'^L a_j) \& (a_k <^L a_j \text{ iff } a_j <'^L a_k)$,

A: $R^{(a_i, a_j)} = F^{(a_i, a_j)}(p^{(a_i, a_j)})$, for every pair $(a_i, a_j) \in Ax A$, where $R^{(a_i, a_j)}$, $F^{(a_i, a_j)}$ and $p^{(a_i, a_j)}$ are the contraction of the social preference ordering, the social welfare function (i.e. the social decision function), and the p profile to the pair (a_i, a_j) .

It seems, that the conditions used in Arrow's impossibility theorem for a 'social welfare function' are formally similar to the conditions of Theorem 4, or are the same of the condition of this theorem. As it is mentioned above we can set a multicriteria decision situation to a voting situation. Then \prec_i is interpreted as the preference order for the i th individual or voter. The Arrowian axioms, and the condition of universal domain and independence from irrelevant alternatives are the same as the conditions of theorem of the lexicographic decision. The difference is that while Arrow's theorem uses strong Pareto concept, the theorem of lexicographic decision method uses weak Pareto concept, and Arrow adds the condition that no individual shall be a 'dictator'. The main result of Arrow's theorem, to be shown that all conditions other than the nondictatorship condition imply that some individual is a dictator.

By deleting specific references to dictators and replacing the weak Pareto concept with the strong Pareto concept, as in Theorem 4., we derive a hierarchy of 'dictators' $\sigma(1), \sigma(2), \dots, \sigma(n)$, which verifies the existence of lexicographic preferences.

References

- [1] Behringer F.A., Lexmaxmin in fuzzy multiobjective decision making, *Optimization* 21 (1990) 23-49.
- [2] Boussou D., Some remarks on the notion of compensation in MCDM, *European Journal of Operation Research*, 26 (1986) 150-160.
- [3] Dombi J., A general framework for the utility/based and outranking methods, *Fuzzy Logic and Soft Computing, Advances in Fuzzy Systems - Application and Theory*, Ed.: B. Bouchon-Meunier, R.R. Yager, R.A. Zadeh, World Scientific, (1995) 202-208.
- [4] Dombi J., Vincze N.J., Universal characterisation of non-transitive preferences, *Mathematical Social Sciences* 27 (1994) 91-104.
- [5] Fishburn P.C., Axioms for Lexicographic Preferences, mimeographed (1972).
- [6] Fishburn P.C., Lexicographic orders, utilities and decision rules: a survey, *Management Science* 11 (1974) 1442-1471.
- [7] Fishburn P.C., Noncompensatory preferences. *Synthese* 33 (1976) 393-403.
- [8] Fishburn P.C., Lexicographic additive differences, *Journal of Mathematical Psychology* 21 (1980) 191-218.

- [9] Fishburn P.C., The foundations of expected utility, D. Reidel Publishing Company, Dordrecht:Holland, Boston:USA, London:England, 1982.
- [10] Fortemps P., Pirlot M., Conjoint axiomatization of Min, DiscriMin and LexiMin (under preparation)
- [11] Frisch A., Hnich B., Kiziltan Z., Miguel I., Walsh T., Global Constraints for Lexicographic Orderings, Ed.:P. Van Heytenryck, Springer-Verlag Berlin Heidelberg (2002) 93-102.
- [12] Hwang CL, Lin MJ., Group Decision Making under Multiple Criteria. Springer Verlag: Berlin Heidelberg, 1987.
- [13] Hwang CL, Lin MJ., Fuzzy Multiple Attribute Decision Making. Springer Verlag: Berlin Heidelberg, 1992.
- [14] Luce R.D., Lexicographic tradeoff structures Theory and Decision 9 (1978) 187-193.
- [15] May K.O., Intransitivity, Utility, and Aggregation of Preference Patterns, Econometrica, 22 (1954) 1-13.
- [16] Olson D.L., Decision Aids for Selection Problems, Springer Verlag: New York, 1996.
- [17] Plott C.R., Little J.T., Parks R.P., Individual Choice When Objects have Ordinal Properties, Social Science Working Paper Number 14. California Institute of Technology 1972.
- [18] Rapcsák T., Többszemponú döntési problémák, AHP módszertan, MTA SZ-TAKI, 2003.
- [19] Solymosi T., Ordinális mérési skálák lexikografikus aggregálása, Adat, modell, elemzés, szerk.:Kovács E., Aula, Budapest, (2001) 119-126.
- [20] Temesi J., A döntéselmélet alapjai, Aula, 2002.
- [21] Marchant T., Towards a theory of MCDM: Stepping away from social choice theory, Mathematical Social Sciences, Elsevier, 45 (2003) 343-363.

Received November, 2004

Minimal inter-particle distance in atom clusters*

Tamás Vinkó†

Abstract

A general method for obtaining minimal interatomic distance in molecule conformation problems is introduced. The method can be applied to a wide family of potential energy functions having reasonable properties. Using this method new lower bounds for the minimal inter-particle distance for the optimal Lennard-Jones and Morse potential functions are derived which are independent from the number of atoms. Improved linear lower bounds for the optimal function values for Lennard-Jones and Morse potentials are also given.

1 Introduction

Given a cluster of n atoms, define $x_i \in \mathbb{R}^3$ ($i = 1, \dots, n$) as the center of the i th atom. The potential energy of the cluster $x = (x_1, \dots, x_n) \in \mathbb{R}^{3n}$ is defined as the sum of the two-body inter-particle pair potentials over all of the pairs, i.e.,

$$E(x) = \sum_{i < j} v(r_{ij}), \quad (1)$$

where $r_{ij} = \|x_i - x_j\|$ and $v(r)$ is the value of a pair potential of distance r . For the pair potential $v(r)$ we set the following requirements to be satisfied:

- (P1) The function v is continuous.
- (P2) There exists a unique s with $v(s) < 0$ and if $r \neq s$ then $v(r) > v(s)$ (single stable state property).
- (P3) If $r \leq s$ then v is strictly decreasing and $v(r) \geq r^{-4}$.
- (P4) If $r > s$ then v is strictly increasing and $v(r) \geq -r^{-4}$.

The properties (P3) and (P4) come from sphere packing arguments, used in the paper. We should use here Cr^{-3} bounds instead, but the *a priori* determination of the constant C is quite difficult.

*This work has been supported by the grants OTKA T 048377 and AÖU 600ü6.

†Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary email: tvinko@inf.u-szeged.hu

The aim of the paper is to obtain lower bounds for the minimal interatomic distance in the optimal structure of (1), independent of the number of atoms and assuming only that the pair potential minimally satisfies properties (P1)–(P4). Many papers deal with this topic, however, they specialized the pair potential function.

1.1 Previous results

The first paper is by Xue *et al.* [9], where a poor lower bound for the minimal distance in Lennard-Jones cluster is established. They also proved that the global optimum can be bounded from below and above by linear (in the number of atoms) functions. In a paper of Maranas and Floudas [7] results for the minimal distance can be found. They established bounds as functions of the number of atoms. That value is useful only for small n since it goes to zero as the number of atoms grows. In another work of Xue [11], a lower bound for inter-particle distance in the optimal Lennard-Jones cluster is given which is independent of the number of atoms in the cluster. Improved lower bound is obtained by Blanc [1]. For Morse clusters (for which property (P3) does not hold) Locatelli and Schoen [5] establish lower bound for the interatomic distance in the optimal structures. In this paper, better lower bounds for the Lennard-Jones and the Morse cluster (where we use the results from [5]) are derived as applications of the introduced general method.

Apart from the theoretical interest, this kind of results can be used efficiently in the construction of global optimization methods, especially in branch-and-bound type methods. As shown by Locatelli and Schoen in [4], information about the minimal interatomic distance can be used efficiently in starting point generator algorithm for (stochastic) optimization methods. Such a lower bound can also be applied to construct special data structures for fast procedures to compute potential functions with large number of atoms, see [12].

1.2 Notation

In the rest of the paper the following notation will be used. The set of real numbers, positive real number and nonnegative integers are denoted by \mathbb{R} , \mathbb{R}_+ and \mathbb{N}_0 , respectively. V denotes the set of functions $v : \mathbb{R}_+ \rightarrow \mathbb{R}$ satisfying properties (P1)–(P4). Using this notation $v \in V$ is supposed in this paper. The global minimizer of the function E is the configuration $x^* \in \mathbb{R}^{3n}$ with

$$E(x^*) = \min_{x \in \mathbb{R}^{3n}} E(x). \quad (2)$$

The global minimum will be denoted by

$$E^* = E(x^*).$$

Let r_{ij} be the Euclidean distance of the points x_i^* and x_j^* ($i, j = 1, \dots, n$). Define the potential energy of particle i as

$$E_i(x) = \sum_{i \neq j} v(\|x_i - x_j\|) \quad (i = 1, \dots, n)$$

and $E_i^* = E_i(x^*)$. It is obvious that

$$E(x) = \frac{1}{2} \sum_{i=1}^n E_i(x) \quad (3)$$

holds. The minimal inter-particle distance in the optimal structure is

$$r^* = \min_{i,j} r_{ij} \quad (i, j = 1, \dots, n). \quad (4)$$

Lower bound for the minimal distance is denoted by q , i.e., our task is to find a good underestimation

$$q \leq r^*.$$

In order to obtain good lower bound q we assume that in the configuration taken into account the minimal distance between the particles equal to q .

The positive root of v is denoted by t . Properties (P1)–(P4) imply that t is unique and $t < s$. Note that with the general method only such a lower bound can be obtained which satisfies $q < t$.

Without loss of generality let us suppose that $x_1 = 0$ and $0 = r_1 < r_2 \leq \dots \leq r_n$, where

$$r_j = \|x_j - x_1\| = \|x_j\| \quad (j = 1, \dots, n).$$

In the rest of the paper we consider only the cases $n > 2$.

2 Lower bound on the minimal inter-particle distance

To give a good lower bound for the minimal inter-particle distance we generalize the arguments given by Xue in [11] and Blanc in [1]. To do that, first we establish an upper bound for E_i^* ($i = 1, \dots, n$). Suppose that $p \in \mathbb{R}_+$ is a parameter such that

$$pq \geq s. \quad (5)$$

Then we use the partition

$$E_1^* = \sum_{q \leq r_j < pq} v(r_j) + \sum_{r_j \geq pq} v(r_j) \quad (6)$$

and give underestimations for the two terms. With suitable chosen parameters we show that if the minimal distance is too small, then we get a contradiction with the upper bound for E_1^* .

2.1 The auxiliary bounds

Lemma 1. *In the optimal configuration the potential energy of particle i is always less than the global minimum of v , i.e. the inequality $E_i^* < v(s)$ holds for all $i = 1, \dots, n$.*

Proof. Let $k = n$ if $i \neq n$ and $k = n - 1$ if $i = n$, and define the configuration $z = (z_1, \dots, z_n)$ in such a way that $z_j = x_j^*$ for all $j \neq i$, $\|z_i - z_k\| = s$ and $\|z_i - z_l\| \geq s$ for all $l \neq i$. Then put the atom z_i to the line determined by the origin point and the coordinates of z_k in such a way that then z_i has the maximal r_j value. Thus $E_i(z) < v(s)$. By construction of z ,

$$E^* - E_i^* = E(z) - E_i(z).$$

Since $E_i(z) < v(s)$ and

$$E^* - E_i^* = E(z) - E_i(z) > E(z) - v(s),$$

we find $E_i^* < v(s)$. □

Lemma 2. For $\frac{q}{2} < a < b$, the index set $\mathcal{J}_{ab} = \{j \mid a \leq r_j < b\}$ has size

$$|\mathcal{J}_{ab}| \leq \left(\frac{2b+q}{q}\right)^3 - \left(\frac{2a-q}{q}\right)^3.$$

Proof. We may assume that the particles are centers of disjoint open balls of radius $q/2$. The cardinality of the set \mathcal{J}_{ab} can not exceed the number of balls with radius $q/2$ that can be contained in the ball centered at the origin with radius $b + q/2$. With volume comparison this gives the upper bound

$$|\mathcal{J}_{ab}| \leq \left(\frac{b + \frac{q}{2}}{\frac{q}{2}}\right)^3.$$

On the other hand, since $r_j \geq a$, we can drop out all the balls with radius $q/2$ from the ball centered in the origin and having radius $a - q/2$. □

Lemma 3. If $pq \geq s$, then the first term of (6) can be underestimated with

$$\sum_{q \leq r_j < pq} v(r_j) \geq v(q) + v(s) ((2p+1)^3 - 1). \quad (7)$$

Proof. Suppose that $r_2 = r_3 = \dots = r_{m+1} = q$, (i.e. there are $m \geq 1$ distances equal to q). Since they give positive contributions we can cancel all of them but one (about what we supposed that exists, see Section 1.2) and this one can be taken out from the sum. Thus

$$\sum_{q \leq r_j < pq} v(r_j) \geq v(q) + \sum_{q < r_j < pq} v(r_j) \quad (8)$$

holds. Moreover, using Lemma 2 and the monotonicity property of the pair potential v we get

$$v(q) + \sum_{q < r_j < pq} v(r_j) \geq v(q) + v(s) \left(\left(\frac{2pq+q}{q}\right)^3 - \left(\frac{2q-q}{q}\right)^3 \right) \quad (9)$$

$$= v(q) + v(s) ((2p+1)^3 - 1). \quad (10)$$

□

Lemma 4. *Let $s \leq pq = R_0 < R_1 < R_2 < \dots$ be an infinite strictly increasing sequence and define the index set $\mathcal{I}_k = \{j \mid 2 \leq j \leq n, R_k \leq r_j < R_{k+1}\}$ ($k = 0, 1, 2, \dots$). If $pq \geq s$, then the second term of (6) can be underestimated with*

$$\sum_{r_j \geq pq} v(r_j) \geq \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k) ((2R_{k+1} + q)^3 - (2R_k - q)^3). \quad (11)$$

Proof. Again, we can use the monotonicity property of v and Lemma 2 with the index set \mathcal{I}_k :

$$\sum_{r_j \geq pq} v(r_j) = \sum_{k=0}^{\infty} \sum_{r_j \in \mathcal{I}_k} v(r_j) \quad (12)$$

$$\geq \sum_{k=0}^{\infty} \sum_{r_j \in \mathcal{I}_k} v(R_k) \quad (13)$$

$$\geq \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k) ((2R_{k+1} + q)^3 - (2R_k - q)^3), \quad (14)$$

which completes the proof. \square

2.2 The general method

Using the above lemmas the following method can be introduced to obtain the minimal interatomic distance in the optimal potential energy function E . Recall that t and s are the zero and the minimizer of the pair potential v , respectively. Suppose that $v \in V$. In Lemma 4 we use an increasing sequence R_k which represents an infinite sequence of spherical shells. Instead of this sequence one can use function $R : \mathbb{R}_+ \times \mathbb{N}_0 \rightarrow \mathbb{R}_+$ having the properties

$$R(Q, k) < R(Q, k+1) \text{ and } R(Q, 0) = c,$$

where $c \in \mathbb{R}_+$ is a constant (in the proof of Lemma 4 this constant is pq , the starting point of the infinite sequence). For technical reasons we use the notation R_k^Q for the functions $R(Q, k)$. Moreover, we write

$$U_c^Q := \{R_k^Q \mid R_k^Q < R_{k+1}^Q \text{ and } R_k^Q = c \text{ and } k = 0, 1, \dots\}.$$

Let us define now

$$F(q, p) := v(q) + v(s) ((2p+1)^3 - 1), \quad (15)$$

$$S(q, p, R) := \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k^Q) \left((2R_{k+1}^Q + q)^3 - (2R_k^Q - q)^3 \right), \quad (16)$$

$$G(q, p, R) := F(q, p) + S(q, p, R). \quad (17)$$

Using these functions and Lemma 3 and 4, we have the lower bound:

$$\begin{aligned} E_1^* &= \sum_{q \leq r_j < pq} v(r_j) + \sum_{r_j \geq pq} v(r_j) \\ &\geq G(q, p, R) \end{aligned} \quad (18)$$

where $p \in \mathbb{R}_+$ such that $pq \geq s$ and $R \in U_{pq}^Q$.

Theorem 1. *Define the function $g_v(q, p, Q) := G(q, p, R)$. If $g_v(q, p, Q) > -\infty$ then in the optimal atom cluster problem (2) the minimal inter-particle distance is greater than or equal to the solution q of the nonlinear system of equations*

$$\frac{\partial g_v(q, p, Q)}{\partial p} = 0, \quad (19)$$

$$\frac{\partial g_v(q, p, Q)}{\partial Q} = 0, \quad (20)$$

$$g_v(q, p, Q) - v(s) = 0. \quad (21)$$

Proof. The finiteness of g_v comes from properties (P3) and (P4). These properties also guarantee that g_v is monotone in q on the interval $[0, s]$. Thus (21) has exactly one solution.

From Lemma 1 we know $E_1^* < v(s)$. Moreover, $g_v \leq E_1^*$ comes from (18). We are looking for the largest q for which the underestimation $g_v < v(s)$ does not hold. Now let us consider the optimization problem

$$\begin{aligned} \max \quad & q \\ \text{s.t.} \quad & g_v(q, p, Q) \geq v(s). \end{aligned} \quad (22)$$

Thus (19) and (20) are the first order optimality conditions for p and Q , respectively, in the optimization problem (22). Finally, (21) guarantees the largest possible q for which the inequality $g_v < v(s)$ does not hold. In this manner the minimal inter-particle distance in (2) is at least q . \square

One can improve the result can be achieved with Theorem 1. If we substitute the first m term of the sequence R_k with variables p_1, \dots, p_m then we have a function G with $m + 2$ variables. Namely,

$$\begin{aligned} G(q, p_1, \dots, p_m, R) &:= F(q, p) + \sum_{i=1}^{m-1} v(p_i q) ((2p_{i+1} + 1)^3 - (2p_i - 1)^3) \\ &\quad + \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k^Q) \left((2R_{k+1}^Q + q)^3 - (2R_k^Q - q)^3 \right), \end{aligned}$$

where $F(q, p)$ is defined in (15), $p_1 q \geq s$, and $R_k^Q \in U_{p_m q}^Q$.

Corollary 1. Define the function $g_v(q, p_1, \dots, p_m, Q) := G(q, p_1, \dots, p_m, R)$. If $g_v > -\infty$ then in the optimal atom cluster problem (2) the minimal inter-particle distance is greater than or equal to the solution q of the nonlinear system of equations

$$\begin{aligned} \frac{\partial g_v(q, p_1, \dots, p_m, Q)}{\partial p_1} &= 0, \\ &\vdots \\ \frac{\partial g_v(q, p_1, \dots, p_m, Q)}{\partial p_m} &= 0, \\ \frac{\partial g_v(q, p_1, \dots, p_m, Q)}{\partial Q} &= 0, \\ g_v(q, p_1, \dots, p_m, Q) - v(s) &= 0. \end{aligned}$$

3 Linear lower bounds on the optimal values

Using the results of the previous section we can establish linear lower bounds for the optimal objective function value. These bounds are valid for arbitrary large clusters.

3.1 The general method

Theorem 2. If q is a lower bound obtained by the usage of Corollary 1 for the minimal inter-particle distance in the problem (2), then there exists a constant K such that

$$-\frac{K}{2}n \leq E^*.$$

Moreover, K can be computed using the value of q .

Proof. Let $i \in \{1, \dots, n\}$ arbitrary but fixed. Recall from Section 1.2 that s is the minimizer and t is the positive root of v , respectively. Let us define the interval $M = [t, pq)$, where $pq \geq s$. Then one can make the underestimation

$$\sum_{\substack{j=1 \\ j \neq i}}^n v(r_{ij}) \geq \sum_{\substack{j=1 \\ j \neq i, r_{ij} \in M}}^n v(r_{ij}) + \sum_{\substack{j=1 \\ j \neq i, r_{ij} \geq pq}}^n v(r_{ij}).$$

Using Lemma 2, an underestimation of the first term is

$$\sum_{\substack{j=1 \\ j \neq i, r_{ij} \in M}}^n v(r_{ij}) \geq v(s) \left((2p+1)^3 - \left(\frac{2t-q}{q} \right)^3 \right). \quad (23)$$

From Lemma 2 and 4 we have a lower bound for the second term:

$$\sum_{\substack{j=1 \\ j \neq i, r_{ij} \geq pq}}^n v(r_{ij}) \geq \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k^Q) \left((2R_{k+1}^Q + q)^3 - (2R_k^Q - q)^3 \right), \quad (24)$$

where $R_k^Q \in U_{pq}^Q$. (see section 2.2). Moreover, as in Corollary 1 we can extend these considerations with introducing more variables in (24). This leads to the underestimation

$$\begin{aligned} \sum_{\substack{j=1 \\ j \neq i}}^n v(r_{ij}) &\geq v(s) \left((2p+1)^3 - \left(\frac{2t-q}{q} \right)^3 \right) + \\ &\quad + \sum_{l=1}^{m-1} v(p_l r^*) \left((2p_{l+1} + 1)^3 - (2p_l - 1)^3 \right) + \\ &\quad + \frac{1}{q^3} \sum_{k=0}^{\infty} v(R_k^Q) \left((2R_{k+1}^Q + q)^3 - (2R_k^Q - q)^3 \right) \\ &=: -K, \end{aligned}$$

where $p_1 q \geq s$ and $R_k^Q \in U_{p_m q}^Q$. If g_v is finite (see Corollary 1) then the substitution of the solution vector from Corollary 1 guarantees the finiteness of K . Finally, equation (3) yields a linear lower bound for the optimal potential function:

$$-\frac{K}{2}n \leq E^*.$$

□

4 Lennard-Jones clusters

In this section the generalized method introduced in the previous section is applied to the Lennard-Jones function.

In general form the Lennard-Jones pair potential function is

$$v_{\sigma, \epsilon}(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right], \quad (25)$$

where ϵ is the pair well depth and $2^{1/6}\sigma$ is the pair separation at equilibrium. In the global optimization literature the function (25) with reduced units, i.e. $\epsilon = \sigma = 1$ and $s = 2^{1/6}$,

$$v_{1,1}(r) = \frac{4}{r^{12}} - \frac{4}{r^6},$$

or the so-called scaled Lennard-Jones pair potential ($\epsilon = 1$, $\sigma = 2^{-1/6}$, $s = 1$)

$$v_{2^{-1/6},1}(r) = \frac{1}{r^{12}} - \frac{2}{r^6} \quad (26)$$

is investigated. Note that the properties (P1)–(P4) required for the application of the general method are satisfied by (25). The scaled version is plotted in Figure 1.

Using (1) and (25), the Lennard-Jones potential function is defined by

$$E_{\sigma,\epsilon}(x) = \sum_{1 \leq i < j \leq n} v_{\sigma,\epsilon}(\|x_i - x_j\|). \quad (27)$$

In the following minimal distance in the optimal Lennard-Jones cluster is given.

4.1 Minimal distance

Theorem 3. *In the optimal Lennard-Jones atom cluster problem the minimal inter-particle distance is greater than or equal to $2^{1/6}\sigma \cdot 0.6187356774$.*

Proof. The translation between the general and the scaled Lennard-Jones pair potential is

$$v_{\sigma,\epsilon}(r) = \epsilon v_{2^{-1/6},1}(r/s), \quad (28)$$

thus the minimal distance scales with s and the potential scales with ϵ . We give a proof for the scaled version; then the result for the general case is straightforward.

For the sake of simplicity, in the proof we use the notation

$$v(r) = v_{2^{-1/6},1}(r) \text{ and } E = E_{2^{-1/6},1}.$$

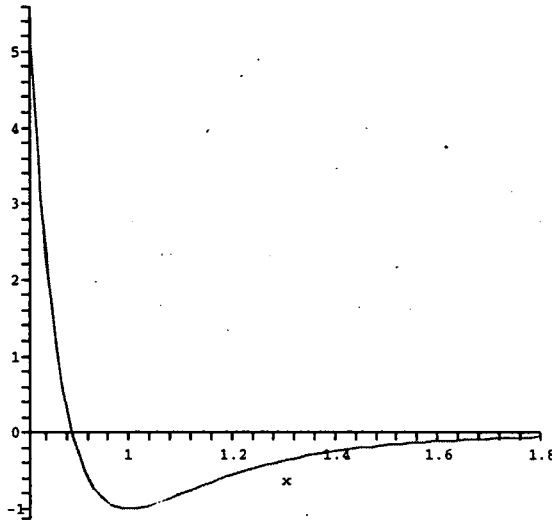


Figure 1: The scaled Lennard-Jones pair potential function.

One can easily see that the zero point and the minimizer point of the function v is

$$t = 2^{-1/6} \quad \text{and} \quad s = 1,$$

respectively.

From Lemma 1 we have $E_1^* < -1$. The lower bound for E_1^* can be established with the usage of Lemma 3 and 4. To prove the theorem by contradiction we should choose a suitable function $R(Q, k)$ to keep that lower bound greater than or equal to -1 .

Define the function $R(Q, k) = pqQ^k$ ($pq \geq 1, Q > 1, k = 0, 1, 2, \dots$). Since property (P4) is satisfied by v , it is easy to see that

$$S_{LJ}(q, p, Q) := \sum_{k=0}^{\infty} \left(\frac{1}{pqQ^{12k}} - \frac{2}{pqQ^{6k}} \right) \left((2pQ^{k+1} + 1)^3 - (2pQ^k - 1)^3 \right) > -\infty \quad (29)$$

holds. Indeed, because $Q > 1$ holds, as k goes to infinity the first term in the sum (i.e. $v(pqQ^k)$) tends to 0 faster than the second term goes to infinity. Thus the function

$$g_v(q, p, Q) := v(q) + 1 - (2p + 1)^3 + S_{LJ}(q, p, Q) \quad (30)$$

is well defined. Figure 2 shows the graph of this function, where the variable $q = 0.618$ is fixed. Note that the function g_v is monotone decreasing in variable q .

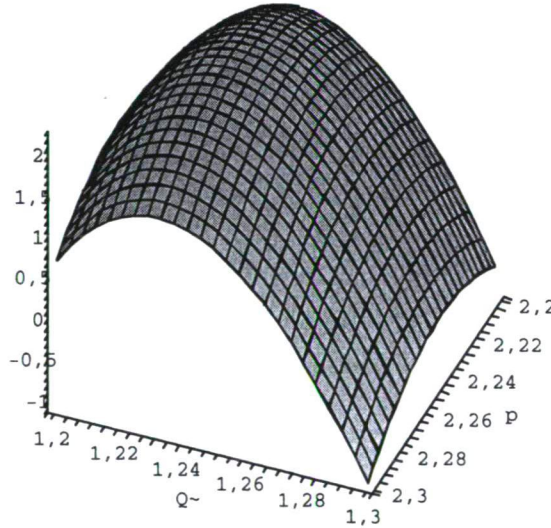


Figure 2: The graph of function $g_v(0.618, p, Q)$.

To obtain a lower bound one has to solve the nonlinear system of equations with three variables:

$$\begin{aligned}\frac{\partial g_v}{\partial p}(q, p, Q) &= 0, \\ \frac{\partial g_v}{\partial Q}(q, p, Q) &= 0, \\ g_v(q, p, Q) + 1 &= 0.\end{aligned}$$

The closed formula of the convergent series (29) and the partial derivative in the nonlinear system of equation above can be calculated with the usage of a symbolic-algebraic system. For this task we used MAPLE 9 [6]. The solution of the nonlinear system is

$$Q = 1.234749976, \quad p = 2.24086158005346, \quad q = 0.61845034503861, \quad (31)$$

which gives a lower bound on the minimal interatomic distance for the optimal scaled Lennard-Jones problem.

As it is stated in Corollary 1, we can improve this bound with introducing more parameters. Using 5 variables instead of 3, one obtains:

$$q = 0.6187356774, \quad (32)$$

which gives a slightly better underestimation for the minimal distance. \square

Note that we do not have significantly better bound with Corollary 1 using more and more variables, but more complicated calculations have to done.

As it is mentioned in the introduction, there are papers about the minimal distance in optimal scaled Lennard-Jones clusters. These results are compared in the following table including the minimal distance obtained in this paper.

Xue [11]	Blanc [1]	general method
0.5	0.6108	0.6187

Note that all these results are independent of the number of particles in the configuration.

The next corollary specializes the previous result for the case of reduced unit.

Corollary 2. *The minimal inter-particle distance in the optimal Lennard-Jones clusters with reduced units is greater than or equal to 0.6945073156.*

4.2 Linear lower bound on the optimal value

Theorem 4. *The optimal Lennard-Jones potential function has the linear lower bound*

$$-138.6775911n \cdot \epsilon \leq E_{\sigma, \epsilon}^* \quad (n = 2, 3, \dots).$$

Proof. One can use the values from the numerical result of Theorem 3 and equation (28) then the statement of the theorem is straightforward from the considerations in section 3.1, thus the proof is omitted. \square

5 Morse clusters

The pair potential function in Morse cluster is

$$v_\rho(r) = e^{\rho(1-r)} \left(e^{\rho(1-r)} - 2 \right), \quad (33)$$

where $\rho > 0$ is a parameter. For $\rho = 6$ the Morse and the scaled Lennard-Jones pair potential are related, they have similar curvature at the minimum point $r = 1$.

Using (33) and (1) the Morse potential function is defined by

$$M_\rho(x) = \sum_{1 \leq i < j \leq n} v_\rho(\|x_i - x_j\|). \quad (34)$$

The zero point and the minimizer point of the function v_ρ is

$$t = 1 - \frac{\ln 2}{\rho} \quad \text{and} \quad s = 1,$$

respectively. Note that if $\rho < \ln 2$ then v_ρ has no positive root. In the context of global optimization, the cases $\rho > 6$ are interesting, since these are more difficult problems than finding the optimal Lennard-Jones structures [2].

5.1 Minimal distance

We must emphasize that property (P3) is not satisfied by the Morse potential. The reason is that the pair potential function v_ρ is defined even in the case $r = 0$, i.e., when two particles are in the same position. In other words the function G from (15) has two roots, i.e. becomes negative for small q values. Thus the general method cannot be applied directly to M_ρ . In this case, information on the minimal inter-particle distances can be helpful. In [5] the minimal inter-particle distance in optimal Morse clusters is investigated. The proposed technique differs from the method introduced by Xue in [11] and from the general method introduced in this paper. In [5] it has been proved that there are positive minimal distances in the optimal Morse clusters for $\rho \geq 6$. Using this information these bounds can be improved by the application of the general method.

In the rest of this subsection we use the notation $M := M_\rho$ for a given $\rho > 0$. From Lemma 1 we know that $M_i^* < -1$ for all $i = 1, \dots, n$ and $\rho > 0$. As for the Lennard-Jones potential, define the function $R(Q, k) := pqQ^k$ ($pq > 1, Q > 1, k = 0, 1, \dots$). The infinite series

$$S_M(q, p, Q) := \sum_{k=0}^{\infty} \left(\left(e^{\rho(1-pqQ^k)} - 1 \right)^2 - 1 \right) \left((2pqQ^{k+1} + 1)^3 - (2pqQ^k - 1)^3 \right) \quad (35)$$

is convergent –the first term of the sum (i.e. $v_\rho(pqQ^k)$) goes to zero faster than the second term goes to infinity–, thus the function

$$g_v(q, p, Q) := v_\rho(q) + 1 - (2p + 1)^3 + S_M(q, p, Q) \quad (36)$$

is well defined.

In Table 1 the results from [5] are collected and compared with the results can be achieved with the usage of the general technique introduced in this paper. Note that the new results are achieved using the results from [5], i.e. using that q must be greater than the second column in Table 1. One can see that the new method produces much better lower bounds, especially for the case $\rho = 6$.

The present method works for $\rho \geq 6$. For $\rho = 5$, the corresponding nonlinear system of equation has no non-negative solution. The technique used in [5] also gives no results for the cases $\rho \leq 6$ (at least without further non-trivial refinements).

5.2 Linear lower bounds on the optimal values

Theorem 5. *The optimal Morse potential function has the linear lower bound for different ρ values:*

$$\begin{aligned}
 -177.6190601n &\leq M_6^* \\
 -97.52208250n &\leq M_7^* \\
 -69.76159670n &\leq M_8^* \\
 -55.71197450n &\leq M_9^* \\
 -47.25499588n &\leq M_{10}^* \\
 -41.61681210n &\leq M_{11}^* \\
 -37.59385566n &\leq M_{12}^* \\
 -34.58070042n &\leq M_{13}^* \\
 -32.24012281n &\leq M_{14}^* \\
 -30.36965466n &\leq M_{15}^*
 \end{aligned}$$

Proof. The values in the statement can be derived by the considerations from section 3.1 and from the numerical result of section 5.1, thus the proof is omitted. \square

ρ	q from [5]	q by the general method
6	0.114	0.4985948046
7	0.376	0.6113121449
8	0.468	0.6796501438
9	0.528	0.7268978345
10	0.574	0.7618207355
11	0.613	0.7887781722
12	0.644	0.8102494106
13	0.672	0.8277671751
14	0.695	0.8423362542
15	0.715	0.8546451536

Table 1: Lower bounds for the minimal distances in optimal Morse clusters for different ρ .

6 Summary

The method introduced in this paper can be used to obtain minimal inter-particle distance in optimal atom clusters. For the usage, only natural requirements are supposed for the pair potential function. Linear lower bounds on the optimal potential energy is also established. As application, new results for the Lennard-Jones and Morse clusters are derived. These theoretical results can be used for accelerating global optimization methods.

Acknowledgment

The author would like to thank Prof. Arnold Neumaier (University of Vienna) for the introduction to this topic and his valuable suggestions and ideas. The author is grateful to the two anonymous referees for the valuable comments and suggestions which have led to better quality in the explanations.

References

- [1] X. Blanc. Lower bounds for the interatomic distance in Lennard-Jones clusters. *Computational Optimization and Applications*, 29:5-12, 2004.
- [2] J.P.K. Doye, R.H. Leary, M. Locatelli and F. Schoen. The global optimization of Morse clusters by potential energy transformations. *INFORMS Journal On Computing*, 16:371-379, 2004.
- [3] H.X. Huang, P. Pardalos and Z.J. Shen. Equivalent formulations and necessary optimality conditions for the Lennard-Jones problem. *Journal of Global Optimization*, 22:97-118, 2002.
- [4] M. Locatelli and F. Schoen. Fast global optimization of difficult Lennard-Jones clusters *Computational Optimization and Applications*, 21:55-70, 2002.
- [5] M. Locatelli and F. Schoen. Minimal interatomic distance in Morse-clusters. *Journal of Global Optimization*, 22:175-190, 2002.
- [6] The Maplesoft Product Site, <http://www.maplesoft.com>
- [7] C. Maranas and C. Floudas. A global optimization approach for Lennard-Jones microclusters. *Journal of Chemical Physics*, 97:7667-7678, 1992.
- [8] J.A. Northby. Structure and binding of Lennard-Jones clusters: $13 \leq n \leq 147$. *Journal of Chemical Physics*, 87:6166-6178, 1987.
- [9] G. L. Xue, R. S. Maier, and J. B. Rosen. Minimizing the Lennard-Jones potential function on a massively parallel computer. *Proceedings of the 6th international conference on Supercomputing*, 409-416, 1992.

- [10] G.L. Xue. Improvements on the Northby algorithm for molecular conformation: better solutions. *Journal of Global Optimization*, 4:425-440, 1994.
- [11] G.L. Xue. Minimum inter-particle distance at global minimizers of Lennard-Jones clusters. *Journal of Global Optimization*, 11:83-90, 1997.
- [12] G.L. Xue. An $O(n)$ time hierarchical tree algorithm for computing force field in n -body simulations. *Theoretical Computer Science*, 197:157-169, 1998.

Received July, 2004

Synthesis of the synchronization of general pipeline systems *

Balázs Ugron[†], Szabolcs Hajdara[†], and László Kozma[†]

Abstract

The pipeline systems and different subtypes of pipelines are interesting parts of parallel systems in software engineering. That is why it seems to be worth dealing with the possibilities of the specification of the synchronization of these systems.

Different methods exist that can be used to synthesize the synchronization of parallel systems based on some kind of specification, but these methods cannot be applied directly for pipeline systems because of some special properties of the pipeline systems and the methods themselves.

The method that seems to be the most promising is the method of Attie and Emerson, which is a synthesization method for many similar processes based on a special temporal logic specification.

In this paper we give an extension of this method so that the extended method will be able to handle more properties of parallel systems, especially of pipeline systems. We will consider not only linear [8], but general pipeline systems too. Furthermore, we give an abstract synchronization of a general pipeline system.

Categories and Subject Descriptors: D.2.1 [Software engineering]: Requirements specification; F.3.1 [Logic and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs – *assertions, invariants*; F.4.1 [Mathematical Logic]: Temporal logic.

Key Words and Phrases: semantic tableaux, pipeline, synthesis, parallel systems, temporal logic.

1 Introduction

In the following, we will consider the synchronization possibilities of a special part of parallel systems, the pipeline systems. As usual (see [1, 2]), we consider only the synchronization part of the processes, because the real computation code usually can be separated from the synchronization part of parallel systems.

*This research work was supported by GVOP-3.2.2-2004-07-005/3.0.

[†]Department of Software Technology and Methodology, Eötvös Loránd University, Pázmány P. sétány 1/c, H-1117 Budapest, Hungary. E-mail: balee@elte.hu, sleet@elte.hu, kozma@ludens.elte.hu

A pipeline system is a parallel system of processes, which is built in order to solve some kind of problems. In the case of the simplest pipeline system, which is linear, the processes are aligned in a row by the connections between them, so every process in the system has exactly two connections, except the first and the last processes, which have only one. The processes between the two ends work on similar tasks, so their synchronization is obviously similar, too.

In this paper, we will consider not only linear, but much more general pipeline systems. In paper [8] we described the synthesis of a linear pipeline system, while in this article general pipeline systems are synthesized. We have only the following assumptions:

1. There are some processes, which have only one connection, which is an output connection. These processes generate the data.
2. There are some processes, which have only one connection, which is an input connection. These processes receive the result.
3. The processes inside the pipeline (that is, which are not data generator or receiver processes) are similar in terms of synchronization.

There are methods in the literature, which can be used to synthesize the synchronization part of a system from temporal logic specification, but these methods cannot be directly applied in this case. For example, the method of Emerson and Clark [2] suffers from the so-called state explosion problem [1], so it cannot be applied for a large number of processes, in practice. Another example is the method of Attie and Emerson [1], which can handle large systems, but this method can be only used for systems consisting many similar processes, and this is not that case.

In this article, after a short description of the synchronization of many similar processes [1], we will introduce an extension of the method, with which it will be possible to handle the case of pipeline systems too.

2 Synthesis of many similar processes

In this section we review the parts in Attie and Emerson's paper [1] that are most important to understand this paper. The reader will generally find only informal definitions in this section, the exact definitions can be found in [1].

First, Attie and Emerson's method specifies that the processes must be similar. In this case, similarity means that any two processes can be exchanged with each other, except their indexes. This restriction is used many times in the method.

2.1 CTL*

The specification language is an extension of the temporal logic CTL*, which is a propositional branching-time temporal logic. The basic modalities of CTL* consist of a path quantifier, either *A* (for all paths) or *E* (for some path) followed by a linear-time formula, which is built up from atomic propositions, the Boolean

operators \wedge , \vee , \neg , and the linear-time modalities G (always), F (sometime), X_j (strong nexttime), Y_j (weak nexttime) and U (until). CTL* formulas are built up from atomic propositions, the Boolean operators \wedge , \vee , \neg , and the basic modalities.

Let us consider the intuitive meaning of the formulas mentioned above. Formula Ef means that there is some maximal path for which f holds; formula Af means that f holds of every maximal path; formula $X_j f$ means that the immediate successor state along the maximal path under consideration is reached by executing one step of process P_j , and formula f holds in that state; formula fUg means that there is some state along the maximal path under consideration where g holds, and f holds at every state along this path at least the previous state.

The usual abbreviations for logical disjunction, implication and equivalence can be introduced easily. Furthermore, some additional modalities as abbreviations can be introduced: $Y_j f$ for $\neg X_j \neg f$, Ff for $trueUf$, Gf for $\neg F \neg f$.

The reader can find the formal definition of the semantics of CTL* formulas in [1].

2.2 The interconnection relation

The interconnection scheme between processes is given by the *interconnection relation* I . $I \subseteq \{i_1, \dots, i_K\} \times \{i_1, \dots, i_K\}$, and $i I j$ iff processes i and j are interconnected. I is a symmetric and irreflexive relation.

Those process pairs that are in the interconnection relation will be synchronized with each other, while the others will not. This means that the behaviour of the system can be simply changed by the interconnection relation. For example, the synchronization of the eating philosophers problem is the same as for the standard mutual exclusion problem – except the interconnection relation.

2.3 MPCTL*

An MPCTL* (Many-Process CTL*) formula consists of a spatial modality followed by a CTL* state formula. A spatial modality is of the form \bigwedge_i or \bigwedge_{ij} . \bigwedge_i quantifies the process index i , which ranges over $\{i_1, \dots, i_K\}$. \bigwedge_{ij} quantifies the process indexes i, j , which range over the elements of I .

The definition of truth in structure M at state s of formula q is given by $M, s \models q$ iff $M, s \models q'$, where q' is the CTL* formula obtained from q by viewing q as an abbreviation and expanding it like

- $M, s \models \bigwedge_i f_i$ iff $\forall i \in \{i_1, \dots, i_K\} : M, s \models f_i$
- $M, s \models \bigwedge_{ij} f_{ij}$ iff $\forall (i, j) \in I : M, s \models f_{ij}$

2.4 The method

In this section, we give an extremely short informal description of the synthesis method of Attie and Emerson, which will be informative enough to catch the point, though.

First, the behaviour of the system needs to be specified in the above described temporal logic language, MPCTL*. This specification is applied to an arbitrary process or process pair from the system.

Any known method (for example the one described in [2]) can be used to synthesize the synchronization skeleton of an arbitrary process pair from the system.

In this way, the method takes advantage of the fact that the processes in the system are similar and produces a global synchronization skeleton for the whole system based on the skeleton synthesized for the pair system.

3 Synthesis of a pipeline system

Our main goal in this paper is to develop a method, with which the synchronization skeleton of a pipeline system can be synthesized. The method of Attie and Emerson which we roughly described above cannot be applied directly in the case of a pipeline system.

The first reason is that the processes in a pipeline system are not similar. Although the processes except the sender and receiver ones of the pipeline are similar, the mentioned two processes differ from them, because they have different state sets from the other processes.

The second reason is that the communication in a pipeline system has a direction – from the sender to the receiver processes. This method does not permit us to distinguish the processes even in the specification, so we cannot handle directions, and the synthesized synchronization code will not be efficient.

First, we give an extension of the method, with which the side processes can be handled too. We introduce one more abstraction level in the method: we separate the processes inside the pipeline from the processes at the ends, handle them as an embedded system, and synthesize the synchronization for them. Finally, we handle the embedded system as a part of a new system, besides the processes at the ends of the pipeline.

3.1 The embedded system

First we give a straightforward solution to the synchronization of the previously mentioned embedded system in the pipeline. This is a very inefficient approach, but in this case the method of Attie and Emerson can be applied directly. In fact, this is the solution of the standard mutual exclusion problem.

In this case, the processes will have three states, a normal (N), a trying (T) and a critical (C) state. A process enters its trying state, when it wants to go to the critical state, and two interconnected processes cannot be in their critical state at the same time. The processes do the communication (data receiving and passing) and the real computation, too, in the critical state.

The synchronization skeleton for such a system is deduced in [1]. The resulted automata can be seen in Figure 1.

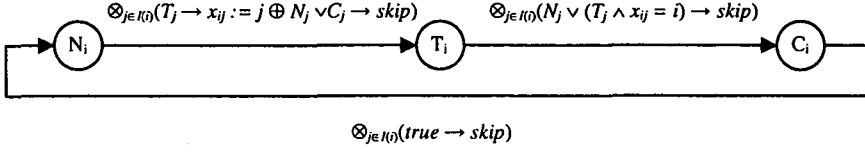


Figure 1: Synchronization skeleton of the embedded system

As we said, this approach is extremely inefficient, because the neighbours of a process cannot do anything, while the process is in critical state, although theoretically they would be able to do the computational part of their work simultaneously.

A much better approach will be shown later, in section 3.2.

3.2 Another approach for the embedded system

The method introduced in section 3.1 is not really applicable for pipeline systems. In this section, the method of Attie and Emerson will be extended so that it could handle such problems.

We realised that the main problem in the synthesization of this part of the system is that the method does not allow us to make a distinction between processes and we cannot express the direction of the communication, so the result will be inefficient.

To get over this issue, we introduce a new definition for the spatial operators defined by Attie and Emerson – or in other words, we define two new spatial operators.

The original definition of the spatial operators can be found in section 2.3. We add a ρ predicate parameter to the spatial operators:

- $M, s \models \bigwedge_i (\rho) f_i$ iff $\forall i \in \{i_1, \dots, i_K\} : \rho \rightarrow M, s \models f_i$
- $M, s \models \bigwedge_{ij} (\rho) f_{ij}$ iff $\forall (i, j) \in I : \rho \rightarrow M, s \models f_{ij}$

This definition intuitively means that a connection between two processes which are defined in the interconnection relation may be actual or non actual in different situations and the actuality of the interconnection is driven by the predicate ρ .

For the sake of effectiveness, there are two critical sections for every process in this approach: a critical section for reading the data from the previous process, and another one for sending the data to the next process. Moreover, there will be a *sent* and a *received* state for each process, because the communication works through shared variables, and the flow of the communication should be driven by the synchronization.

The processes will have many states: N (normal), T (try to read), R (read), C (check), W (work), E (try to send), S (send) and finally A (after send). The two critical states are R and S , and the restriction is that if a process is in its state S , then the following process must not be in its state R , and vice versa.

Let us see what happens in these states. State N is the start state of every process. State T is a trying state before the R critical section, which is for reading. State C is a checkpoint after the reading. State W is the state in which the process does its real computation work. State E is a trying state before the S critical section, which is for sending. Finally, state A is another checkpoint, now after sending.

Let us see the extended MPCTL* specification of the synchronization of the embedded system:

1. Initial state (every process is initially in its normal state):

$$\bigwedge_i N_i$$

2. It is always the case that any move P_i makes from its N state is into its T state, and such a move is always possible (and similarly for the states R , W and S):

$$\bigwedge_i AG(N_i \Rightarrow (AY_i T_i \wedge EX_i T_i))$$

$$\bigwedge_i AG(R_i \Rightarrow (AY_i C_i \wedge EX_i C_i))$$

$$\bigwedge_i AG(W_i \Rightarrow (AY_i E_i \wedge EX_i E_i))$$

$$\bigwedge_i AG(S_i \Rightarrow (AY_i A_i \wedge EX_i A_i))$$

3. It is always the case that any move P_i makes from its T state is into its R state – but such a move is not definitely possible (and similarly for the states C , E and A):

$$\bigwedge_i AG(T_i \Rightarrow AY_i R_i)$$

$$\bigwedge_i AG(C_i \Rightarrow AY_i W_i)$$

$$\bigwedge_i AG(E_i \Rightarrow AY_i S_i)$$

$$\bigwedge_i AG(A_i \Rightarrow AY_i N_i)$$

4. P_i is always in exactly one state of the state set:

$$\bigwedge_i AG(N_i \equiv \neg(T_i \vee R_i \vee C_i \vee W_i \vee E_i \vee S_i \vee A_i))$$

$$\bigwedge_i AG(T_i \equiv \neg(N_i \vee R_i \vee C_i \vee W_i \vee E_i \vee S_i \vee A_i))$$

$$\bigwedge_i AG(R_i \equiv \neg(N_i \vee T_i \vee C_i \vee W_i \vee E_i \vee S_i \vee A_i))$$

$$\begin{aligned}
 & \bigwedge_i AG(C_i \equiv \neg(N_i \vee T_i \vee R_i \vee W_i \vee E_i \vee S_i \vee A_i)) \\
 & \bigwedge_i AG(W_i \equiv \neg(N_i \vee T_i \vee R_i \vee C_i \vee E_i \vee S_i \vee A_i)) \\
 & \bigwedge_i AG(E_i \equiv \neg(N_i \vee T_i \vee R_i \vee C_i \vee W_i \vee S_i \vee A_i)) \\
 & \bigwedge_i AG(S_i \equiv \neg(N_i \vee T_i \vee R_i \vee C_i \vee W_i \vee E_i \vee A_i)) \\
 & \bigwedge_i AG(A_i \equiv \neg(N_i \vee T_i \vee R_i \vee C_i \vee W_i \vee E_i \vee S_i))
 \end{aligned}$$

5. Liveness: if P_i is in state T , then some time it will reach state R (and similarly for the states C , E and A):

$$\begin{aligned}
 & \bigwedge_i AG(T_i \Rightarrow AFR_i) \\
 & \bigwedge_i AG(C_i \Rightarrow AFW_i) \\
 & \bigwedge_i AG(E_i \Rightarrow AFS_i) \\
 & \bigwedge_i AG(A_i \Rightarrow AFN_i)
 \end{aligned}$$

6. A transition by a process cannot cause a transition by another one:

$$\begin{aligned}
 & \bigwedge_{ij} AG((N_i \Rightarrow AY_j N_i) \wedge (N_j \Rightarrow AY_i N_j)) \\
 & \bigwedge_{ij} AG((T_i \Rightarrow AY_j T_i) \wedge (T_j \Rightarrow AY_i T_j)) \\
 & \bigwedge_{ij} AG((R_i \Rightarrow AY_j R_i) \wedge (R_j \Rightarrow AY_i R_j)) \\
 & \bigwedge_{ij} AG((C_i \Rightarrow AY_j C_i) \wedge (C_j \Rightarrow AY_i C_j)) \\
 & \bigwedge_{ij} AG((W_i \Rightarrow AY_j W_i) \wedge (W_j \Rightarrow AY_i W_j)) \\
 & \bigwedge_{ij} AG((E_i \Rightarrow AY_j E_i) \wedge (E_j \Rightarrow AY_i E_j)) \\
 & \bigwedge_{ij} AG((S_i \Rightarrow AY_j S_i) \wedge (S_j \Rightarrow AY_i S_j)) \\
 & \bigwedge_{ij} AG((A_i \Rightarrow AY_j A_i) \wedge (A_j \Rightarrow AY_i A_j))
 \end{aligned}$$

7. Data flow control: a process in state T waits for the previous process to reach state A and a process in state C waits for the previous process to leave A (and similarly the two other rules):

$$\bigwedge_{ij} (j < i) AG((T_i \wedge \neg A_j) \Rightarrow \neg EX_i true)$$

$$\bigwedge_{ij} (j < i) AG((C_i \wedge A_j) \Rightarrow \neg EX_i true)$$

$$\bigwedge_{ij} (i < j) AG((E_i \wedge C_j) \Rightarrow \neg EX_i true)$$

$$\bigwedge_{ij} (i < j) AG((A_i \wedge \neg C_j) \Rightarrow \neg EX_i true)$$

8. Always there is a possible step:

$$AGEX true$$

If the set of the process-indices is $\{1, 2\}$ (so the processes are P_1 and P_2), then we get the specification of a pair-system. From this specification we can synthesize the synchronization skeleton of the pair-system with the method of Emerson and Clarke [2]. We implicitly applied the method on the parametric spatial operators introduced by us. In the following, the non trivial steps of the synthesis can be seen. Only the main cases are considered, because the other cases can be done by the analogy of the following ones. Note that the dashed lines in the figures mean that trivial steps are omitted there.

Figure 2 shows how the blocks of the initial node can be constructed.

In Figure 3 the construction of the titles of the result of the previous step can be seen.

Figure 4 shows an example of the case when one of the processes has an eventually condition, but because of the parameters of the spatial operators none of the processes has to be blocked.

After this there are a lot of similar steps as Figure 5 shows.

Figure 6 illustrates an example of making blocks of a node where one of the processes has to wait for the another, and Figure 7 shows the titles of the result of this step (only P_1 can execute the changing of its state).

In Figure 8 an example is shown of the case when both processes have the possibility of blocking, but only one of them has to wait for the other. In Figure 9 the titles of the result of the previous step can be seen.

The \downarrow sign in the tableau means that the relevant branch of the tableau is unsatisfiable, so this branch has to be eliminated.

Based on this tableau we can construct the global state transition diagram, which can be seen in Figures 16-20 in Appendix A.

Based on the global state transition diagram, we can construct the DAGs (see Figure 21 in Appendix B) and the fragments (see Figure 22 in Appendix B) of every

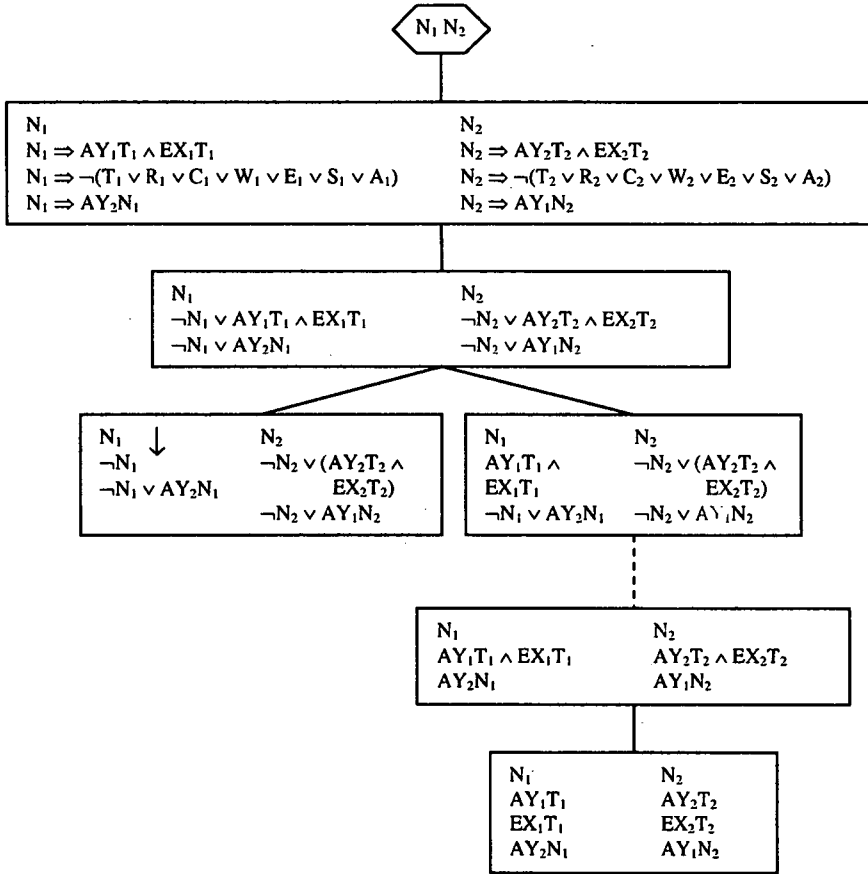


Figure 2: Blocks of the initial node

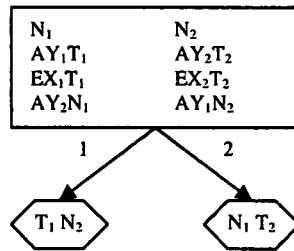


Figure 3: Titles of the result set of the blocks of the initial node that can be seen in Figure 2

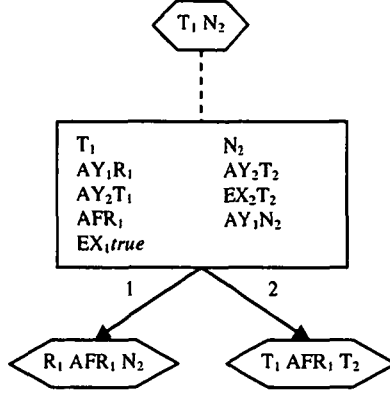


Figure 4: None of the processes blocks but one of them has an eventually condition

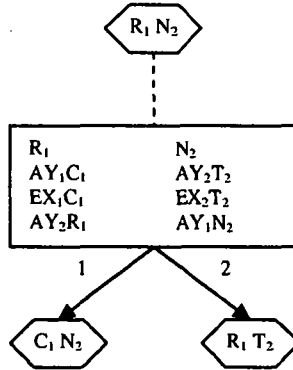


Figure 5: There are a lot of steps similar to step one and two

node. Based on the fragments, the model can be constructed shown in Figure 23-24 in Appendix B.

From the model we can construct the final deterministic automata for all processes. If the first process (P_1) is in its state N_1 then the second process (P_2) can be in all of its states except state R_2 , and P_1 has the possibility to step in all of this states. So the condition of the transition of P_1 from state N_1 to state T_1 is $\neg R_2$. The conditions of the transitions in states T_1 , R_1 , C_1 , W_1 are the same. If P_1 is in state E_1 and P_2 is in state C_2 , then P_1 cannot step, so the condition of the transition from state E_1 to state S_1 is $\neg R_2 \wedge \neg C_2$. The conditions of the other transitions can be determined similarly. Figure 10 shows the result, the synchronization skeleton for P_1 and P_2 .

Note that it cannot happen that P_1 is in state N_1 and P_2 is in state R_2 , so the

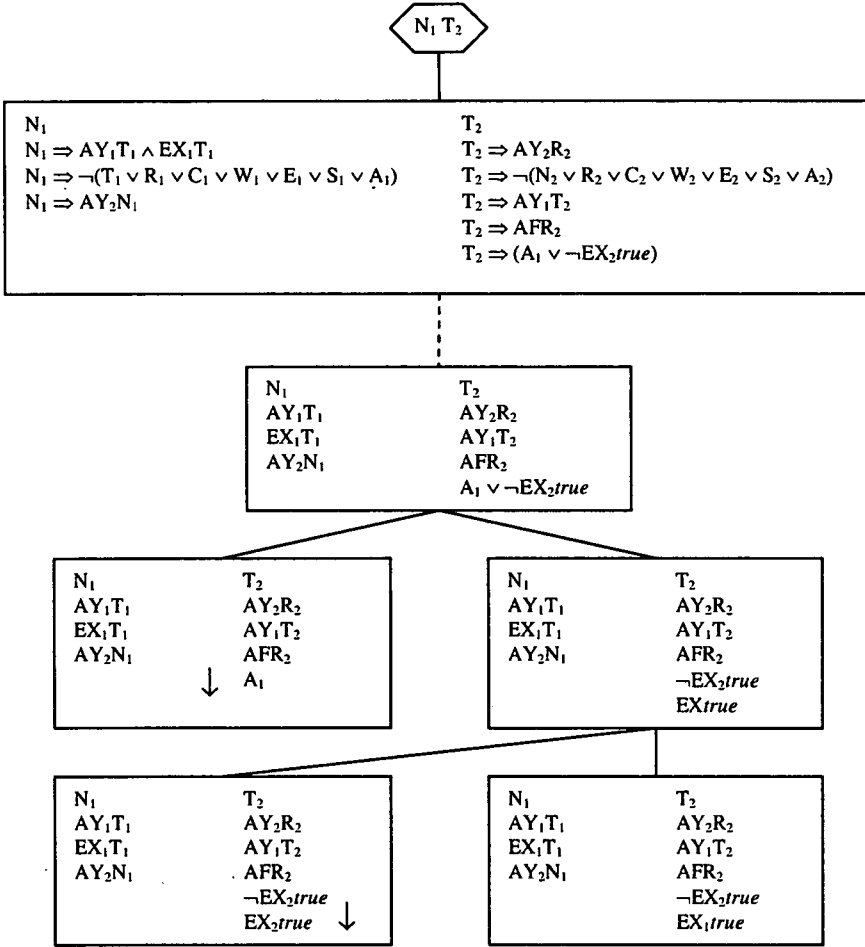


Figure 6: Example of generating blocks of a node where the processes has to wait

$\neg R_2$ in condition of the transition from N_1 to T_1 can be eliminated. Similarly, we can do this with all of the transitions. The simplified synchronization skeletons can be seen in Figure 11.

From this synchronization skeleton we can generate the synchronization code for every process with the method of Attie and Emerson [1]. The finite deterministic automata resulted by the method can be seen in Figure 12.

Note that in the case of this synchronization, nothing keeps a process from working – that is, stepping in its state W – while the neighbours are working, so the processes can really work in parallel in this case.

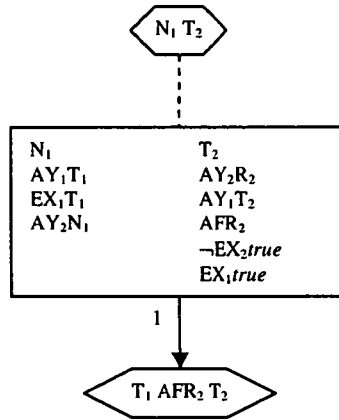


Figure 7: Titles of the result of the previous step can be seen in Figure 6

3.3 The new three-process system

As we said, there are special processes at the two ends of the pipeline, which only send and receive data. For the sake of simplicity, let us assume that the sender processes only produce the data and pass them on to the proper process in the embedded system, and similarly, the receiver processes only pick up the processed data from the proper process in the embedded system, and then work with that. These special processes must be handled in a special way.

The sender and receiver processes are similar in the sense that they are connected with only one another process, which they receive data from, or which they send data to. It is enough to consider only one sender and one receiver process when we generate the synchronization skeleton of the whole system, because the synchronization code of the selected sender and receiver process will naturally be suitable for the other sender and receiver processes, and respectively, the synchronization for the processes that are connected to the sender and receiver processes will be reusable, too. That is why from this point on we will consider only one sender and one receiver process in the system.

Now we can look at our process system as a system composed of three processes. The first process is the selected sender, the second is the embedded system and the third is the selected receiver process. We have to build the synchronization skeleton of this system. This system has only three processes, so we can handle it with the method of Emerson and Clarke [2], without running into the state explosion problem.

There is still one more subject that we have to discuss. The middle process in this system is a system of processes itself, which makes the specification of our three-process system quite difficult. We should not just say, for example, that the pseudo process has a state for reading data, because this means that the first

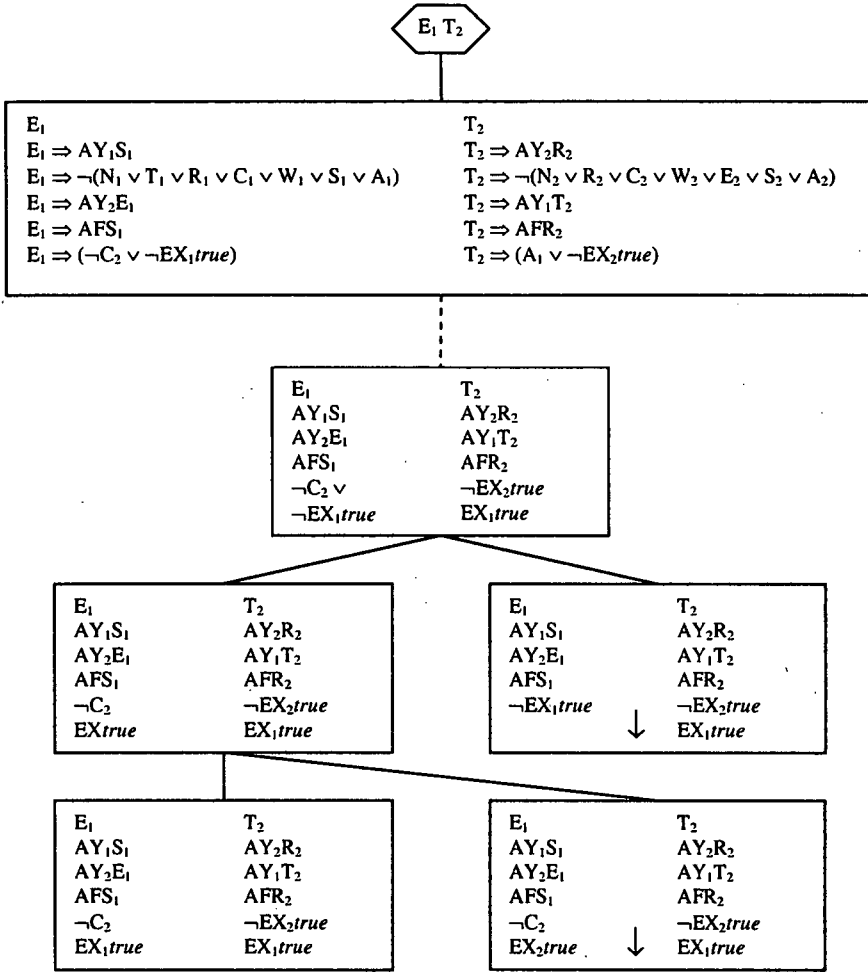


Figure 8: Example of generating blocks of a node where both processes have the possibility to be blocked, but only one of them has to wait

process of the pseudo process reads the data, and at the same time, the last process theoretically can send data, which means that the whole pseudo process sends data to the receiver process, too. As a result, the pseudo process would be in two states at the same time, which is not allowed.

We give two solutions to this issue.

The first solution is that we handle the pseudo process as two processes – in this case, we have a four-process system instead of a three-process one –, which are independent in the four-process system; one of them is connected to the sender

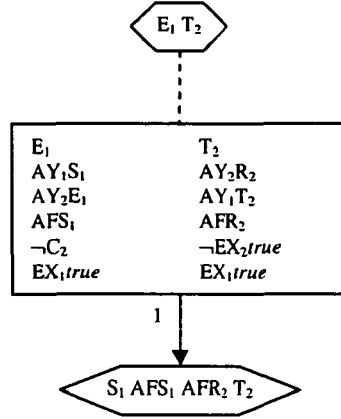


Figure 9: Titles of the result of the previous step can be seen in Figure 7

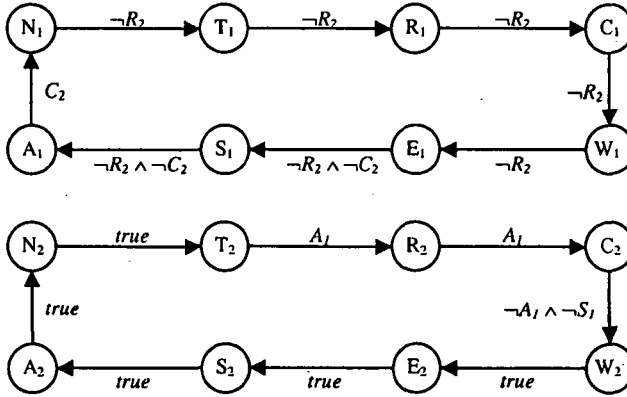


Figure 10: The synchronization skeletons

process, and the other is connected to the receiver. This is a reasonable approach, because there is a hidden connection between the two pseudo processes, and this connection is handled by the synchronization of the embedded system.

The second approach is to define the states of the embedded system as pairs, so we will have state pairs like (N, N) , (N, S) , (R, S) and so on. For example, (N, N) means that the embedded system does not read or send data, while (N, S) means that the system does not read, but sends data and (R, S) means that the system reads and sends data at the same time. With such a type of set of states, we can express the behaviour of the system in a quite efficient way.

The second approach is more complicated than the first one (because of the large number of the states of the system), so we chose the first approach for the

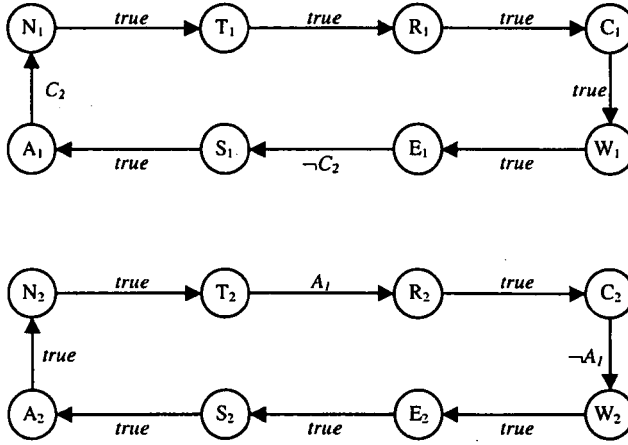


Figure 11: Simplified synchronization skeletons

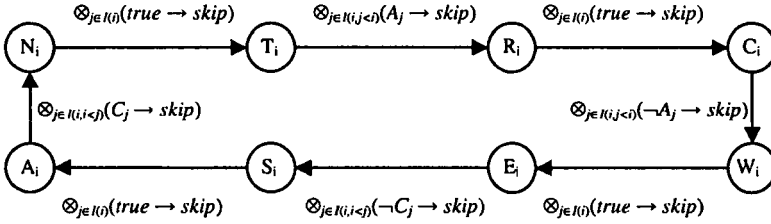


Figure 12: Improved synchronization skeleton of the embedded system

solution. We show only the connection between the sender process and the embedded system. The synchronization of the embedded system and the receiver process can be deduced similarly.

The states of the sender process are:

- J:** normal (working) state,
- K:** try to send state,
- L:** sending state,
- M:** after sending state.

Using these states we can give the temporal logic specification of the system – see Appendix C. For the specification, CTL* was used. Based on this specification, we are able to generate the synchronization skeleton of the system. We used the synthesization method of Emerson and Clarke. The synchronization skeleton for the sender process and the first pseudo process of the embedded system can be

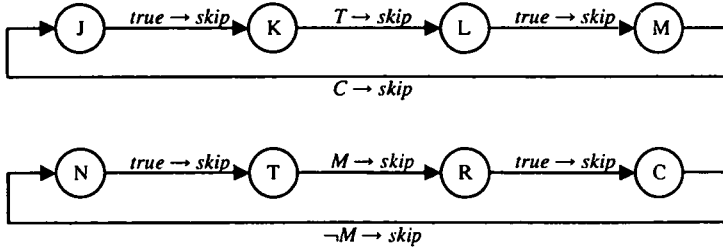


Figure 13: Synchronization skeleton for the sender process and the embedded system

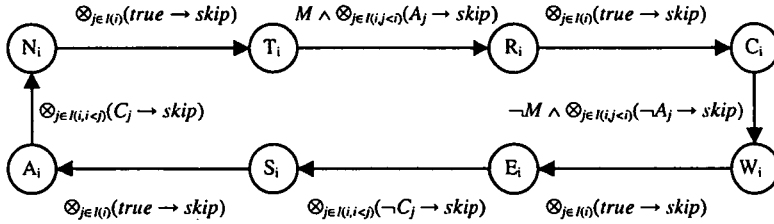


Figure 14: The modified synchronization skeleton for the first process of the embedded system

seen in Figure 13. Finally, the synchronization skeleton of the first and the last processes of the embedded system must be modified properly based on the result in Figure 13 – the transition conditions of the first process of the embedded system will be the conjunction of the original conditions, and the conditions in the proper transitions of the synchronization of the sender process and the pseudo embedded system – see in Figure 14. The new transition conditions for the last process of the embedded system can be deduced similarly.

4 An application

Where could this method be used? There are many complicated processor networks, which can be used for computational purposes; for example, the so-called butterfly network (see [7]). An n -level butterfly network can be constructed in a recursive way, which can be seen in Figure 15. The reason why these processor networks are interesting is that there are many parallel algorithms that can be computed on them, such as the Fast Fourier Transformation on a butterfly network (see [7]).

If we apply our result to an n -level butterfly, then we will have n generator processes, which are connected to the nodes at the left side of the butterfly, and we have n receiver processes, which are connected to the nodes at the right side

of the butterfly; the connections between the other processes can be defined in the relation I , in a proper way for the FFT working on a butterfly network. A proper I relation will be described in the following.

The number of the processes in $B_n = n2^{n-1}$.

Let the numbering of the processes in B_2 be like in Figure 15.

Then the numbering of the processes in B_{n+1} comes from the following rules:

- The numbering of the first B_n component in B_{n+1} is the same as of B_n (i.e.: processes 1 – 4 of B_3 in Figure 15).
- The numbering of the second B_n component in B_{n+1} is the numbering of B_n shifted by $n2^{n-1}$ (i.e.: processes 5 – 8 of B_3 in Figure 15).
- The numbering of the remainder processes (the right side column) is $n2^n + 1 - n2^n + 2^n$ (i.e.: processes 9 – 12 of B_3 in Figure 15).

As a result, the relation I consists of the following pairs:

- The pairs in the two B_n components.
- $\forall i \in [1 \dots 2^{n-1}] : ((n-1)2^{n-1} + i, n2^n + i) \in I$ (i.e.: (3, 9) and (4, 10) of B_3 in Figure 15).
- $\forall i \in [1 \dots 2^{n-1}] : ((n-1)2^{n-1} + i, n2^n + 2^{n-1} + i) \in I$ (i.e.: (3, 11) and (4, 12) of B_3 in Figure 15).
- $\forall i \in [0 \dots 2^{n-1} - 1] : (n2^n - i, n2^n + 2^{n-1} - i) \in I$ (i.e.: (7, 9) and (8, 10) of B_3 in Figure 15).
- $\forall i \in [0 \dots 2^{n-1} - 1] : (n2^n - i, n2^n + 2^n - i) \in I$ (i.e.: (7, 11) and (8, 12) of B_3 in Figure 15).

The synchronization of the communication between the processes are defined in this way, and we do not have to bother with the “business logic” of how the processes compute the data they send to the connected processes.

5 Conclusion

This paper introduced a general pipeline tool, by which a complex application, such as the parallel FFT, can be solved in a short and simple way.

Most of the programs that are working on some kind of data channels (see [5]) can be handled by the method described above. If some modification is still needed, the modification can be restricted to the temporal logic specifications and the relation I , so the above method can be processed by the analogy of the above; moreover, there are different tools exist that can help the process – for instance, the method of Emerson and Clarke [2]; namely, the finite deterministic automata for the pair-system can be generated from the CTL specification automatically, or even

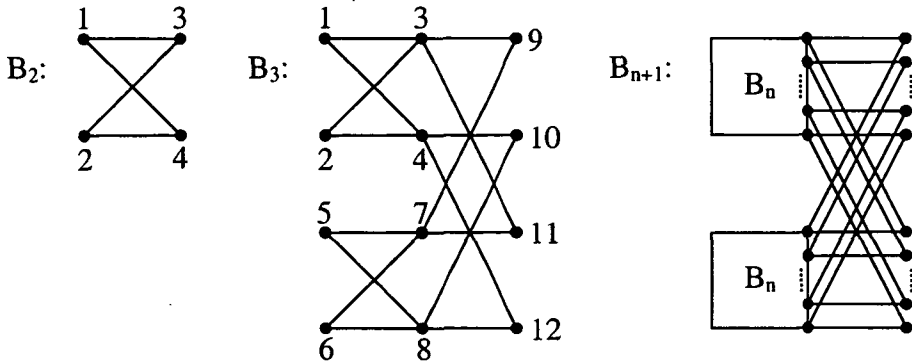


Figure 15: The structure of butterfly processor networks

concrete Java code can be generated with an object-oriented extension (see [4]) of [1] (though a straightforward modification is needed because of the parameterized spatial operators introduced in this paper) etc.

Note, that in the case of FFT no modification was needed, only the proper relation I had to be defined.

6 Future work

The idea of synthesizing the synchronization of pipeline systems comes from the hardware designing of graphics cards. We will work on to meet these demands.

The correctness of the algorithm should be proofed.

An effective tool for deadline checking should be developed.

During the communication, now it is possible that a process has received data but it has to wait until the other processes that receive data from the same sender process receive the data. Theoretically, it would be possible for a data receiver process to step forward in this situation. That is, the data flow control may be improved.

References

- [1] P. C. Attie, E. A. Emerson: *Synthesis of Concurrent Systems with Many Similar Processes*, ACM TOPLAS Vol. 20, No. 1, (January 1998) pp. 51-115
- [2] E. A. Emerson, E. M. Clarke: *Using branching time temporal logic to synthesize synchronization skeletons*, Science of Computer Programming, 2 (1982), pp. 241 - 266
- [3] Sz. Hajdara, L. Kozma, B. Ugron: *Synthesis of a system composed by many similar objects*, Annales Univ. Sci. Budapest., Sect. Comp. 22 (2003)

- [4] Sz. Hajdara, B. Ugron: *An example of generating the synchronization code of a system composed by many similar objects*, 17th European Conference on Object-Oriented Programming (ECOOP), The 13th Workshop for PhD Students in Object-Oriented Systems (2003)
- [5] Z. Hernyák, Z. Horváth, V. Zsók: *Clean-CORBA Interface Supporting Skeletons*, 6th International Conference on Applied Informatics, Eger 2004
- [6] L. Kozma: *A transformation of strongly correct concurrent programs*, Proc. of the Third Hungarian Computer Science Conference 1981, 157-170
- [7] F. T. Leighton: *Introduction to Parallel Algorithms and Architectures*, 1992
- [8] B. Ugron, Sz. Hajdara: *Synthesis of the synchronization of pipeline systems*, 6th International Conference on Applied Informatics, Eger 2004

Appendix A

The following figures (16 – 20) describe the global state transition diagram of the two-process system of the embedded system.

Since the global state transition diagram of the system is too large, we had to split it into five figures. So we used the following notation: the bold box elements mark those elements that can be continued but they are continued in an other figure. Dotted box elements show boxes which can be found in a previous figure, so in the global state transition diagram there is an edge to such boxes.

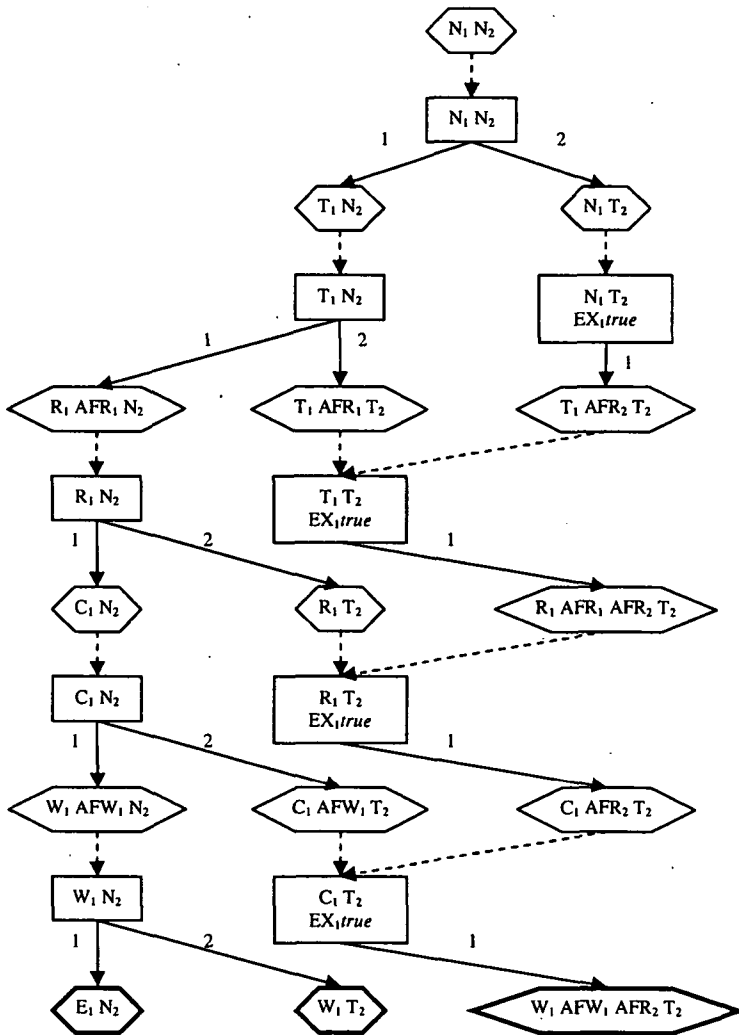


Figure 16: The global state transition diagram (part 1)

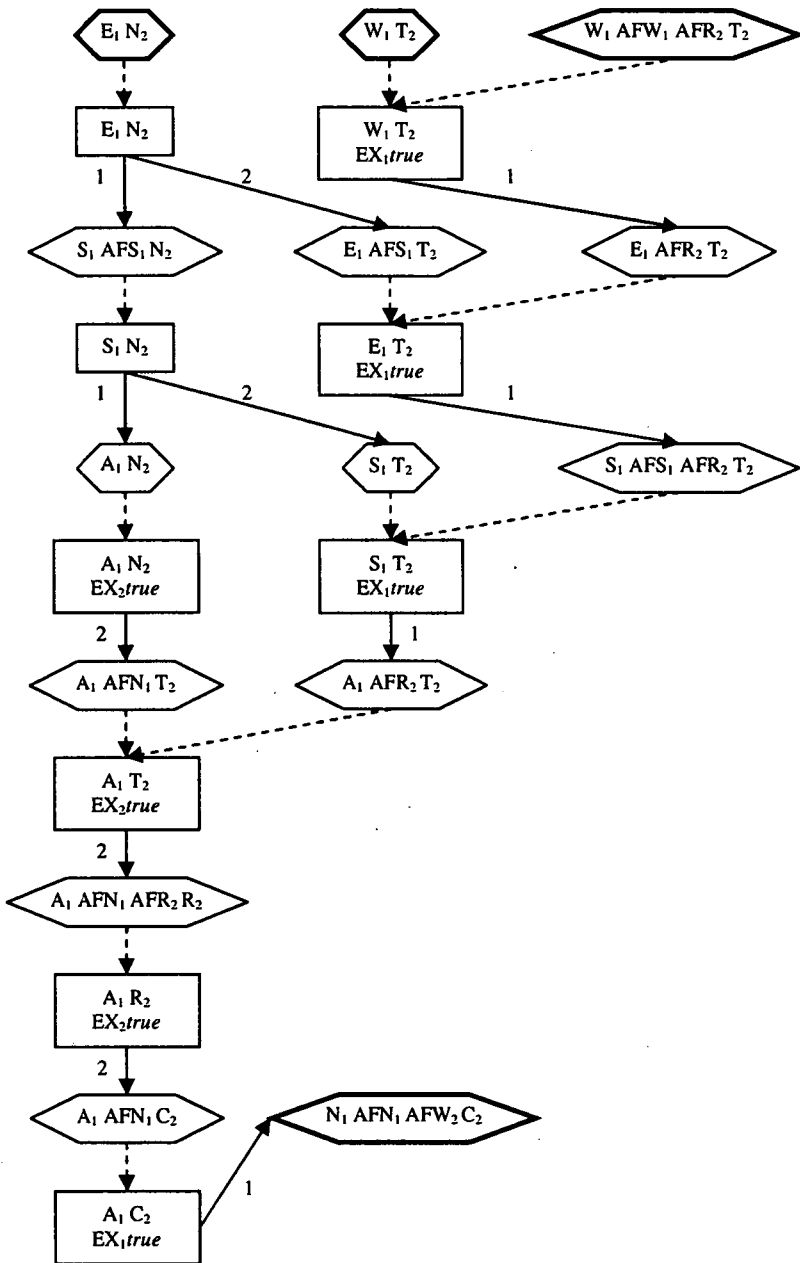


Figure 17: The global state transition diagram (part 2)

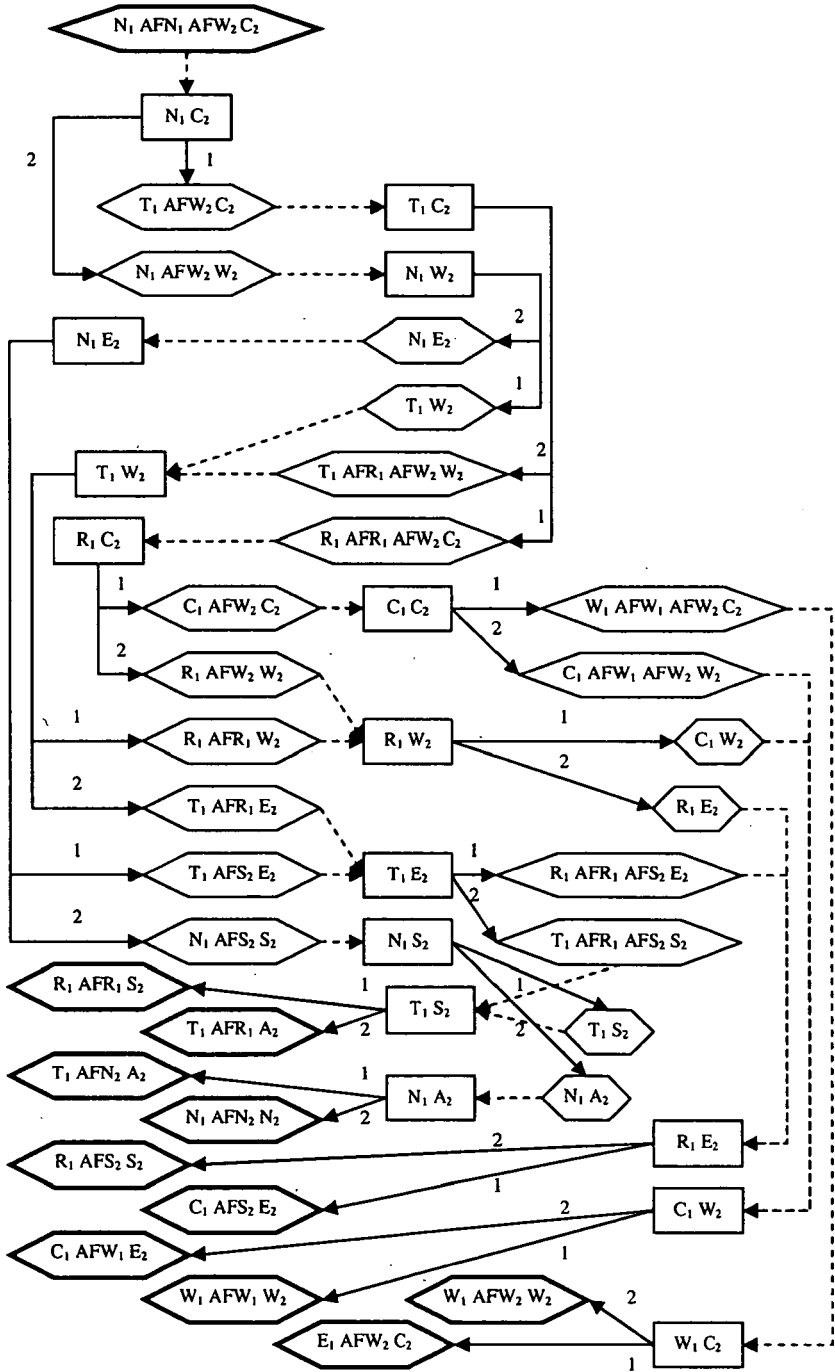


Figure 18: The global state transition diagram (part 3)

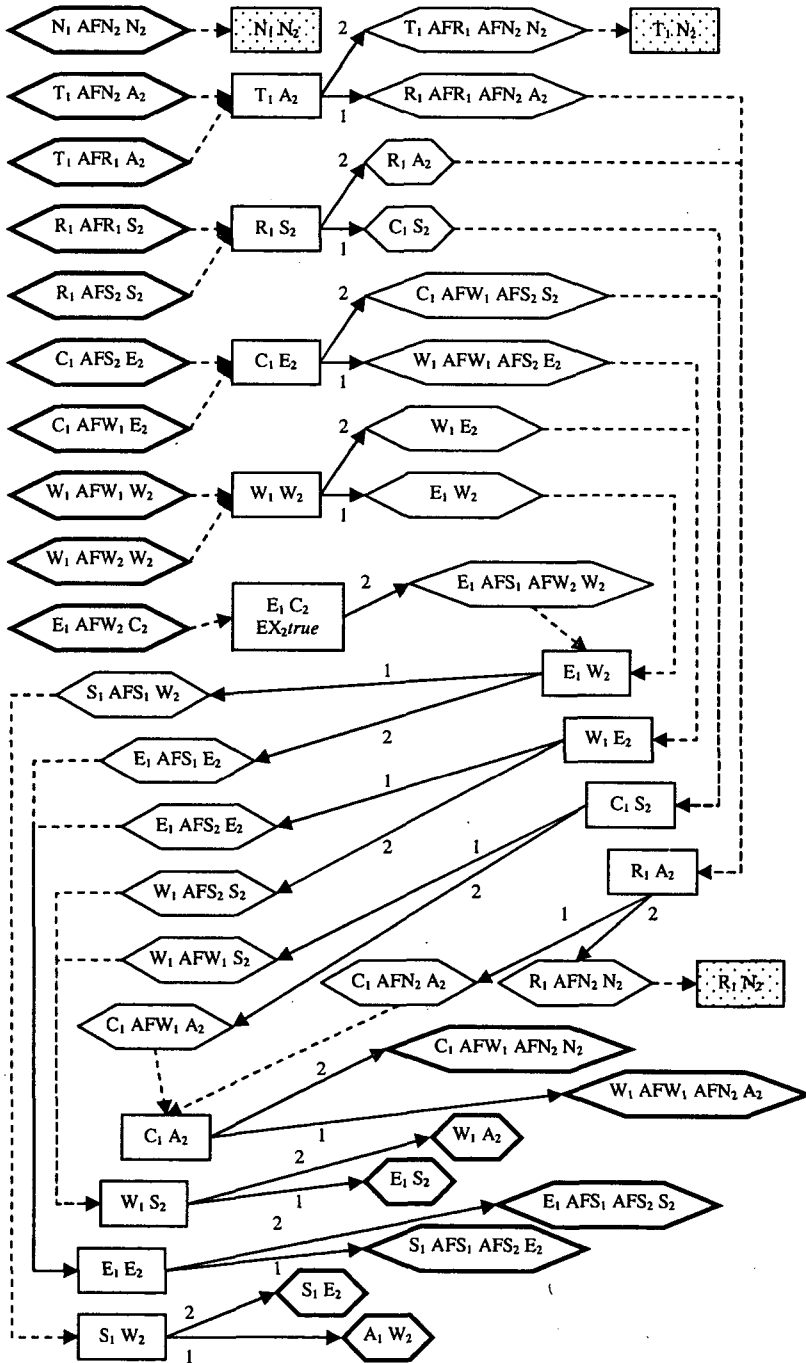


Figure 19: The global state transition diagram (part 4)

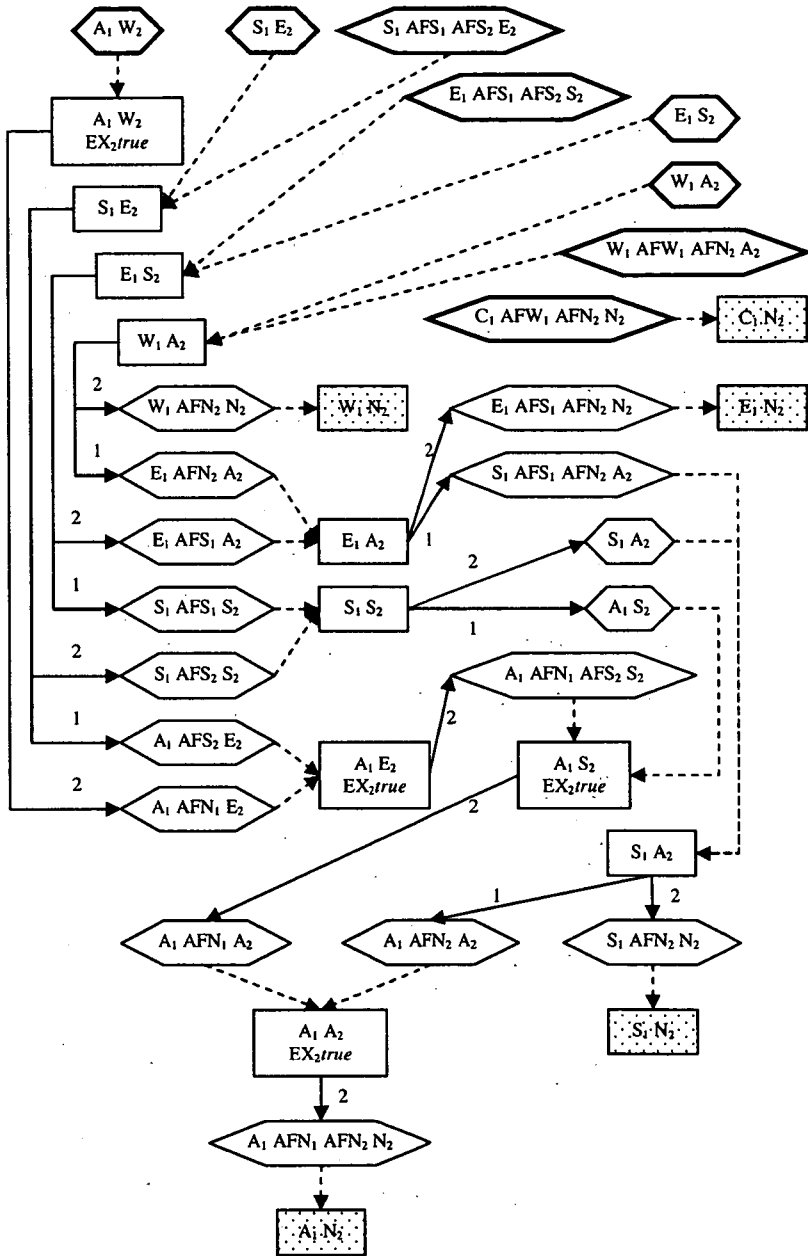


Figure 20: The global state transition diagram (part 5)

Appendix B

The following figures (21 – 24) describe the model of the two-process system of the embedded system.

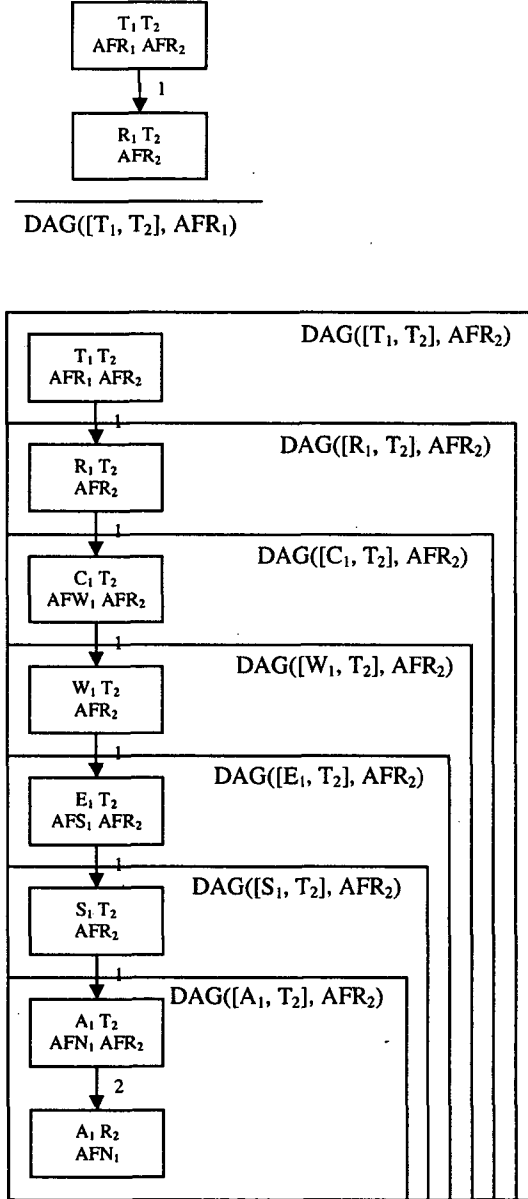


Figure 21: The DAGs that are needed to construct $\text{frag}([T_1, T_2])$

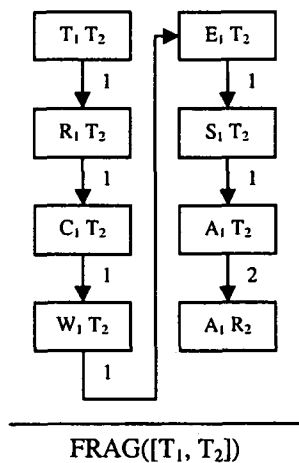


Figure 22: The fragment of the $[T_1, T_2]$ AND node

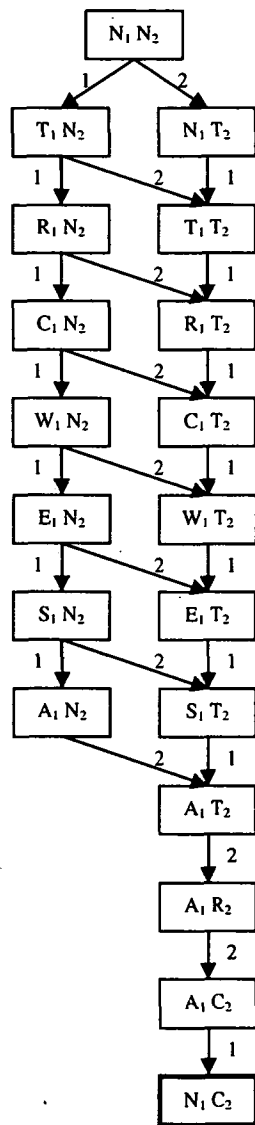


Figure 23: The model of the system

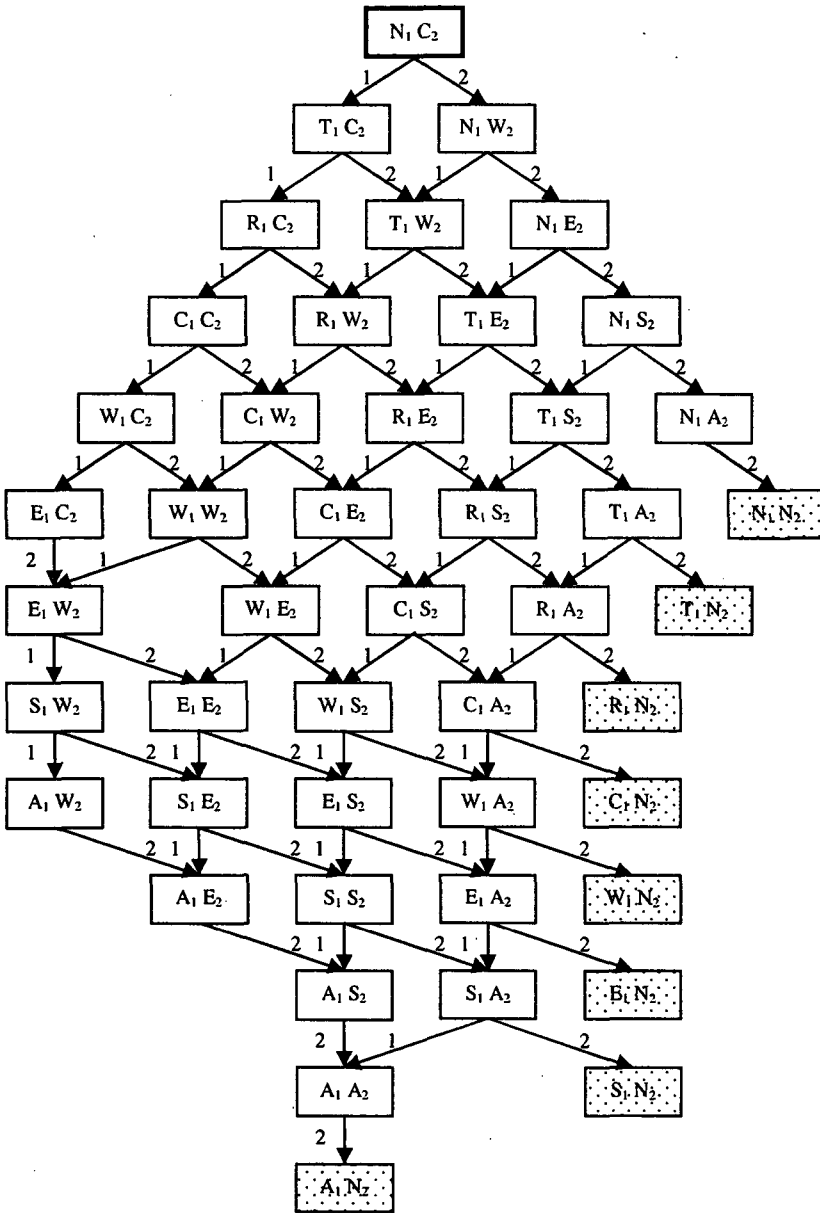


Figure 24: The model of the system

Appendix C

In the following, the temporal logic specification can be seen for the system that is built from the sender process and the embedded system.

For the sake of simplicity, we can join the states N , W , E , S and A of the first process of the embedded system (because the first process only receives data from the sender, the sender will not keep the first process from sending). Let the name of the joined state be N (Normal). Furthermore, we omit the indexes of the states, because the states of the sender process are labeled in another way.

Note that in this case P_1 is the second process and the sender is the first process.

1. Initial state (every process is initially in its normal state):

$$J \wedge N$$

2. It is always the case that any move P_1 makes from its N state is into its T state, and such a move is always possible (and similarly for the state R and for the states J and L of the sender):

$$AG(J \Rightarrow (AY_1K \wedge EX_1K))$$

$$AG(L \Rightarrow (AY_1M \wedge EX_1M))$$

$$AG(N \Rightarrow (AY_2T \wedge EX_2T))$$

$$AG(R \Rightarrow (AY_2C \wedge EX_2C))$$

3. It is always the case that any move P_1 makes from its T state is into its R state – but such a move is not definitely possible (and similarly for the state C and for the states K and M of the sender):

$$AG(K \Rightarrow AY_1L)$$

$$AG(M \Rightarrow AY_1J)$$

$$AG(T \Rightarrow AY_2R)$$

$$AG(C \Rightarrow AY_2N)$$

4. The processes are always in exactly one state of the state set:

$$AG(J \equiv \neg(K \vee L \vee M))$$

$$AG(K \equiv \neg(J \vee L \vee M))$$

$$AG(L \equiv \neg(J \vee K \vee M))$$

$$AG(M \equiv \neg(J \vee K \vee L))$$

$$AG(N \equiv \neg(T \vee R \vee C))$$

$$AG(T \equiv \neg(N \vee R \vee C))$$

$$AG(R \equiv \neg(N \vee T \vee C))$$

$$AG(C \equiv \neg(N \vee T \vee R))$$

5. Liveness: if P_1 is in state T , then some time it will reach state R (and similarly for the state C and states K and M of the sender):

$$AG(K \Rightarrow AFL)$$

$$AG(M \Rightarrow AFJ)$$

$$AG(T \Rightarrow AFR)$$

$$AG(C \Rightarrow AFN)$$

6. A transition by a process cannot cause a transition by another one:

$$AG(J \Rightarrow AY_2J)$$

$$AG(K \Rightarrow AY_2K)$$

$$AG(L \Rightarrow AY_2L)$$

$$AG(M \Rightarrow AY_2M)$$

$$AG(N \Rightarrow AY_1N)$$

$$AG(T \Rightarrow AY_1T)$$

$$AG(R \Rightarrow AY_1R)$$

$$AG(C \Rightarrow AY_1C)$$

7. Data flow control: a process in state T waits for the sender process to reach state M and a process in state C waits for the sender process to leave M (and similarly for the sender):

$$AG((K \wedge \neg T) \Rightarrow \neg EX_1 \text{true})$$

$$AG((M \wedge \neg C) \Rightarrow \neg EX_1 \text{true})$$

$$AG((T \wedge \neg M) \Rightarrow \neg EX_2 \text{true})$$

$$AG((C \wedge M) \Rightarrow \neg EX_2 \text{true})$$

8. Always there is a possible step:

$$AGEX \text{true}$$

Functional Dependencies over XML Documents with DTDs

Sven Hartmann*, Sebastian Link* and Klaus-Dieter Schewe*

Abstract

In this article an axiomatisation for functional dependencies over XML documents is presented. The approach is based on a representation of XML document type definitions (or XML schemata) by nested attributes using constructors for records, disjoint unions and lists, and a particular null value, which covers optionality. Infinite structures that may result from referencing attributes in XML are captured by rational trees. Using a partial order on nested attributes we obtain non-distributive Brouwer algebras. The operations of the Brouwer algebra are exploited in the soundness and completeness proofs for derivation rules for functional dependencies.

Keywords: eXtensible Markup Language, nested attributes, subattributes, rational trees, functional dependencies, axiomatisation

1 Introduction

Over the last decade the eXtensible Markup Language (XML) [5] has attracted a lot of attention in research and practice. Its spectrum of usage spreads from data exchange on the web to a direct use as a data model. In fact, the language shows a lot of similarities to semi-structured data [1] and to object-oriented databases [12].

The treatment of XML as a data model requires re-investigating core problems of database theory. Therefore, it is no surprise that database dependency theory [13] has recently started a revival in the context of XML. The research interest first focused on the classes of keys [4, 11] and functional dependencies [3, 16, 18], which represent the most common and at the same time easiest class of dependencies.

However, the problem is still not completely solved. The major drawback of the work by Arenas, Fan and Libkin and similarly Vincent and Liu is the restriction to a relational representation of XML documents. That is, XML documents are considered as some sets of (generalised) tuples, which then can be treated analogously to the relational model. However, it is possible to formulate functional dependencies on XML documents that are not preserved by the relational representation. In

*Massey Information Science Research Centre, Private Bag 11222, Palmerston North, New Zealand. E-mail: [s.hartmann|s.link|k.d.schewe]@massey.ac.nz

other words, these theories are adequate as long as we only deal with functional dependencies that can be expressed on a relational representation of XML documents. Going beyond this restricted class of functional dependencies requires an extended theory or a different approach to the problem.

Our own work in this area originates from a more classical approach dealing with dependencies in higher-order data models such as the higher-order Entity-Relationship model (HERM) [14] or the object-oriented data model (ODM) [12]. The basic idea is to consider nested attributes that can be built from constructors for records, sets, lists, etc. Furthermore, our first interest is devoted to the logical and mathematical foundations of dependency theory, i.e. we first address problems of axiomatisation, number of possible dependencies, complexity of closure building, etc.

Using just the record- and set-constructors we obtained an axiomatisation in [6], extended in [7]. Additional constructors for lists, multisets and disjoint unions have been handled in [10]. Unfortunately, the presence of the union-constructor, in particular in connection with the set-constructor, requires an extension of the theory to weak functional dependencies, i.e. disjunctions of functional dependencies. Rather astonishingly, among the three “bulk” constructors the list-constructor is the easiest one. So far it is the only part of the theory that could be generalised to multi-valued dependencies [9]. Other work on multi-valued dependencies for XML [17] is again “relationally minded”.

In this article we extend our theory of functional dependencies to XML documents. We show how to represent XML elements by nested attributes. In particular, we represent the Kleene-star by the list-constructor, i.e. we have order and duplicates. In our theory it is also possible to use the multiset- or set-constructor instead, thus neglecting order or duplicates. We may also treat all three “bulk” constructors together. However, as this would blow up the article we made the choice to restrict ourselves to only the easiest of the bulk constructors. A glimpse of the necessary extensions for the other two bulk constructors can be obtained from [10].

In any case the combination of a bulk constructor with the union-constructor is only satisfactory, if some form of restructuring is taken into account. The early work in [2] handles only the set-constructor, but even for this the theory would be equivalent to restricting the union-constructor in a way that it can only occur as the outermost constructor. This is insufficient, if subattributes are considered. Therefore, we use an extended form of restructuring.

However, in order to fully capture functional dependencies over XML documents we have to face two major extensions:

1. We have to consider rational tree attributes, which result from reference structures in XML documents. We will see that the extension arising from this problem is not severe. The major observation is that the subattribute lattice becomes infinite, but this does not affect the derivation of dependencies. Note that all previous work on functional dependencies for XML including [3] neglect references.

2. We have to consider functional dependencies on embedded elements. These dependencies can be “lifted” through the constructors, i.e. they induce functional dependencies on complete XML documents. Such dependencies have not been considered in our previous work. However, the “lifting rules” became already indispensable in the presence of the union-constructor, as this constructor leads to axioms on embedded structures [10].

In other words, this article takes a reasonable fragment of the theory from [10] and extends it with respect to these two problems. The result is a quite uniform axiomatisation for functional documents over XML documents.

In the remainder of the article we first investigate the relationship between XML documents and nested attributes in Section 2. We show how to map the regular expressions in XML document type definitions to the attribute constructors. Furthermore, we extend nested attributes by rational trees and use them to represent the infinite structures that may arise from references in XML documents. We then define a partial order on nested attributes and show that the set of subattributes of a given attribute forms nearly a Brouwer algebra — however, distributivity does not hold.

In Section 3 we introduce functional dependencies and first prove the soundness of some derivation rules for them. These soundness rules imply properties of closures, i.e. sets of subattributes that depend functionally on a given set of subattributes. This leads to the notion of “strong higher-level ideal” or SHL-ideal for short. We show a central theorem about such SHL-ideals, which states that we can always find two values in the associated domain that coincide exactly on a given SHL-ideal. This theorem is indeed central for the proof of the completeness of the derivation rules. The completeness theorem will be the major result of this article.

2 XML and Nested Attributes

In this section we define our extended model of nested attributes including rational tree attributes. We show how to use these attributes to represent XML document type definitions. Finally, we look a bit closer into the structure of sets of subattributes and show that we obtain non-distributive Brouwer algebras.

2.1 Elements in XML and Constructors

The structure of XML documents is prescribed by a document type definition (DTD) [1] or (almost equivalently) by an XML schema. Basically, such a DTD is a collection of element definitions, where each element is defined by a regular expression made out of element names and a single base domain *PCDATA*. Without loss of generality we may assume to have more than one domain. Then we can isolate those element definitions that lead only to domains. These elements can be represented by simple attributes.

Definition 1. A *universe* is a finite set \mathcal{U} together with domains (i.e. sets of values) $\text{dom}(A)$ for all $A \in \mathcal{U}$. The elements of \mathcal{U} are called *simple attributes*.

For all other element definitions we may assume without loss of generality — just spend a few more element names, if necessary — that they are normalised in the sense that they only contain element names and no domains, and they only use exactly one of the constructors for sequences, Kleene-star or alternative.

Then they can be represented as nested attributes as defined next. We use a set \mathcal{L} of labels, and tacitly assume that the symbol λ is neither a simple attribute nor a label, i.e. $\lambda \notin \mathcal{U} \cup \mathcal{L}$, and that simple attributes and labels are pairwise different, i.e. $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 2. Let \mathcal{U} be a universe and \mathcal{L} a set of labels. The set \mathcal{N} of *nested attributes* (over \mathcal{U} and \mathcal{L}) is the smallest set with $\lambda \in \mathcal{N}$, $\mathcal{U} \subseteq \mathcal{N}$, and satisfying the following properties:

- for $X \in \mathcal{L}$ and $X'_1, \dots, X'_n \in \mathcal{N}$ we have $X(X'_1, \dots, X'_n) \in \mathcal{N}$;
- for $X \in \mathcal{L}$ and $X' \in \mathcal{N}$ we have $X[X'] \in \mathcal{N}$;
- for $X_1, \dots, X_n \in \mathcal{L}$ and $X'_1, \dots, X'_n \in \mathcal{N}$ we have $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \in \mathcal{N}$.

We call λ a *null attribute*, $X(X'_1, \dots, X'_n)$ a *record attribute*, $X[X']$ a *list attribute*, and $X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$ a *union attribute*. As record and list attributes have a unique leading label, say X , we often write simply X to denote the attribute.

Thus, a Kleene-star element definition $\langle \text{!ELEMENT } X(Y)^* \rangle$ will be represented by the nested attribute $X[Y]$, a sequence element definition $\langle \text{!ELEMENT } X(Y_1, \dots, Y_n) \rangle$ by $X(Y_1, \dots, Y_n)$, and an alternative element definition $\langle \text{!ELEMENT } X(Y_1 \mid \dots \mid Y_n) \rangle$ by $X(X_1(Y_1) \oplus \dots \oplus X_n(Y_n))$ with some new invented labels X_1, \dots, X_n . Furthermore, as the plus-operator in regular expressions can be expressed by the Kleene-star, an element definition $\langle \text{!ELEMENT } X(Y)^+ \rangle$ will be represented by the nested attribute $X(Y, X'[Y])$ with some new invented label X' . Similarly, optional elements can be expressed as alternatives with empty elements, thus an element definition $\langle \text{!ELEMENT } X(Y?) \rangle$ will be represented by the nested attribute $X(Y) \oplus X'(\lambda)$.

We can now extend the association dom from simple to nested attributes, i.e. for each $X \in \mathcal{N}$ we will define a set of values $\text{dom}(X)$.

Definition 3. For each nested attribute $X \in \mathcal{N}$ we get a *domain* $\text{dom}(X)$ as follows:

- $\text{dom}(\lambda) = \{\top\}$;
- $\text{dom}(X(X'_1, \dots, X'_n)) = \{(X_1 : v_1, \dots, X_n : v_n) \mid v_i \in \text{dom}(X'_i) \text{ for } i = 1, \dots, n\}$ with labels X_i for the attributes X'_i ;
- $\text{dom}(X[X']) = \{[v_1, \dots, v_n] \mid v_i \in \text{dom}(X') \text{ for } i = 1, \dots, n\}$, i.e. each element in $\text{dom}(X[X'])$ is a finite list with elements in $\text{dom}(X')$;

- $\text{dom}(X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)) = \{(X_i : v_i) \mid v_i \in \text{dom}(X'_i) \text{ for } i = 1, \dots, n\}$.

Hence, each element in a DTD can be represented by a nested attribute. An XML document is then represented by a value $v \in \text{dom}(X)$ of the nested attribute X that represents the root. In the following we assume without loss of generality — rename, if necessary — that labels are used only once in a representing nested attribute. In this way we may identify labels with nested attributes labelled by them.

2.2 Attributes in XML and Rational Trees

Besides element definitions a DTD also contains attribute definitions. Attributes are associated with elements. Neglecting some of the syntactic sugar, we basically have three types of attributes:

- attributes with domain *CDATA*, which can be represented again by simple attributes;
- attributes with domain *ID*, which can be ignored;
- attributes with domain *IDREF* or *IDREFS*, which can be replaced by the label, or the list of labels, respectively, of the referenced elements.

More formally, we extend our Definition 2 of nested attributes by adding $\mathcal{L} \subseteq \mathcal{N}$. We say that a label $Y \in \mathcal{L}$ occurring inside a nested attribute X , is a *defining label* iff it is introduced by one of the three cases in Definition 2. Otherwise it is a *referencing label*. We require that each label Y appears at most once as a defining label in a nested attribute X , and that each referencing label also occurs as a defining label. In other words, if we represent a nested attribute by a labelled tree, a defining label is the label of a non-leaf node, and a referencing label is the label of a leaf node.

Using labels we can subsume the attributes of an element in the element definition using a sequence constructor. Attributes with domain *CDATA* will be represented by simple attributes, attributes with domain *IDREF* will be represented by the label of the referenced element, and attributes with domain *IDREFS* will be represented by the list of labels of the referenced elements.

We still have to extend Definition 3. For this assume $X \in \mathcal{N}$ and let Y be a referencing label in X . If we replace Y by the nested attribute that is defined by Y within X , we call the result an *expansion* of X . Note that in such an expansion a label may now appear more than once as a defining label, but all the nested attributes defined by a label can be identified, as the corresponding sets of expansions are identical.

In order to define domains assume set of *label variables* $\psi(Y)$ for each $Y \in \mathcal{L}$. Then for each expansion X' of a nested attribute X we define $\text{dom}(X')$ as in Definition 3 with the following modifications:

- for a referencing label Y we take $\text{dom}(Y) = \psi(Y)$;

- for a label Y defining the nested attribute Y' take $\text{dom}(Y) = \{y : v \mid y \in \psi(Y), v \in \text{dom}(Y')\}$;
- allow only such values v in $\text{dom}(X')$, for which the values of referencing labels also occur inside v exactly once at the position of a defining label.

Finally, define $\text{dom}(X) = \bigcup_{X'} \text{dom}(X')$, where the union spans over all expansions X' of X .

2.3 Subattributes

In classical dependency theory for the relational model we considered the powerset $\mathcal{P}(R)$ for a relation schema R , which is a Boolean algebra with order \subseteq . We have to generalise this for nested attributes starting with a partial order \geq . However, this partial order will be defined on equivalence classes of attributes. We will identify nested attributes, if we can identify their domains.

Definition 4. \equiv is the smallest *equivalence relation* on \mathcal{N} satisfying the following properties:

- $\lambda \equiv X()$;
- $X(X'_1, \dots, X'_n) \equiv X(X'_1, \dots, X'_n, \lambda)$;
- $X(X'_1, \dots, X'_n) \equiv X(X'_{\sigma(1)}, \dots, X'_{\sigma(n)})$ for any permutation σ ;
- $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \equiv X_{\sigma(1)}(X'_{\sigma(1)}) \oplus \dots \oplus X_{\sigma(n)}(X'_{\sigma(n)})$ for any permutation σ ;
- $X(X'_1, \dots, X'_n) \equiv X(Y_1, \dots, Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \equiv X_1(Y_1) \oplus \dots \oplus X_n(Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X[X'] \equiv X[Y]$ iff $X' \equiv Y$;
- $X(X'_1, \dots, Y_1(Y'_1) \oplus \dots \oplus Y_m(Y'_m), \dots, X'_n) \equiv Y_1(X'_1, \dots, Y'_1, \dots, X'_n) \oplus \dots \oplus Y_m(X'_1, \dots, Y'_m, \dots, X'_n)$;
- $X[X_i(X'_i)] \equiv X(X_i[X'_i])$.

Basically, the equivalence definition (apart from the last case) states that λ in record attributes can be added or removed, and that order in record and union attributes does not matter. The last case in Definition 4 covers an obvious restructuring rule, which was already introduced in [2].

In the following we identify \mathcal{N} with the set \mathcal{N}/\equiv of equivalence classes. In particular, we will write $=$ instead of \equiv , and in the following definition we should say that Y is a subattribute of X iff $\tilde{X} \geq \tilde{Y}$ holds for some $\tilde{X} \equiv X$ and $\tilde{Y} \equiv Y$.

Definition 5. For $X, Y \in \mathcal{N}$ we say that Y is a *subattribute* of X , iff $X \geq Y$ holds, where \geq is the smallest partial order on \mathcal{N} satisfying the following properties:

- $X \geq \lambda$ for all $X \in \mathcal{N}$;
- $X \geq X'$ for all expansions X' of X ;
- $X(Y_1, \dots, Y_n) \geq X(X'_{\sigma(1)}, \dots, X'_{\sigma(m)})$ for some injective $\sigma : \{1, \dots, m\} \rightarrow \{1, \dots, n\}$ and $Y_{\sigma(i)} \geq X'_{\sigma(i)}$ for all $i = 1, \dots, m$;
- $X_1(Y_1) \oplus \dots \oplus X_n(Y_n) \geq X_{\sigma(1)}(X'_{\sigma(1)}) \oplus \dots \oplus X_{\sigma(n)}(X'_{\sigma(n)})$ for some permutation σ and $Y_i \geq X'_i$ for all $i = 1, \dots, n$;
- $X[Y] \geq X[X']$ iff $Y \geq X'$;
- $X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)] \geq X(X_1[X'_1], \dots, X_n[X'_n])$;
- $X[X_1(X'_1) \oplus \dots \oplus X_k(X'_k)] \geq X[X_1(X'_1) \oplus \dots \oplus X_\ell(X'_\ell)]$ for $k \geq \ell$;
- $X(X_{i_1}[\lambda], \dots, X_{i_k}[\lambda]) \geq X_{\{i_1, \dots, i_k\}}[\lambda]$.

Obviously, $X \geq Y$ induces a projection map $\pi_Y^X : \text{dom}(X) \rightarrow \text{dom}(Y)$. For $X \equiv Y$ we have $X \geq Y$ and $Y \geq X$ and the projection maps π_Y^X and π_X^Y are inverse to each other.

Note that the last three cases in Definition 5 covers the restructuring for lists of unions, which needs some more explanation. Obviously, if we are given a list of elements labelled with X_1, \dots, X_n , we can take the individual sublists – preserving the order – that contain only those elements labelled by X_i and build the tuple of these lists. In this case we can turn the label into a label for the whole sublist. This explains the third to last subattribute relationship. In case $n = 1$ this is subsumed by the last equivalence in Definition 4.

Using the subattribute relationship for record attributes we obtain

$$X(X_1[Y_1], \dots, X_n[Y_n]) \geq X(X_{i_1}[Y_{i_1}], \dots, X_{i_k}[Y_{i_k}])$$

for $\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$. But then also

$$X[X_{i_1}(Y_{i_1}), \dots, X_{i_k}(Y_{i_k})] \geq X(X_{i_1}[Y_{i_1}], \dots, X_{i_k}[Y_{i_k}])$$

holds as already explained. It is therefore natural to require the second to last property. It just means that a list with elements labelled by X_1, \dots, X_k can be mapped to the sublist – preserving the order – that contains only the elements with labels X_1, \dots, X_ℓ . We may of course take any subset of the labels here, but this is already captured by the possibility to permute the components in a union attribute.

In a list we can also map each element to \top , the unique element in $\text{dom}(\lambda)$. In fact, the subattribute of the form $X[\lambda]$ only counts the number of elements in the list. This is not affected by first separating the list according to labels, so we obtain the last subattribute relationship.

However, restructuring requires some care with labels. If we simply reused the label X in the last property in Definition 5, we would obtain

$$X[X_1(X'_1) \oplus X_2(X'_2)] \geq X(X_1[X'_1], X_2[X'_2]) \geq X(X_1[X'_1]) \geq X(X_1[\lambda]) \geq X[\lambda].$$

However, the last step here is wrong, as the left hand side refers to the length of the sublist containing the elements with label X_1 , whereas the right hand side refers to the length of the whole list, i.e. elements have labels X_1 or X_2 . No such mapping can be claimed. In fact, what we really have to do is to mark the list label in an attribute of the form $X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$ to indicate the inner union attribute, i.e. we should use $X_{\{1, \dots, n\}}$ (or even $X_{\{X_1, \dots, X_n\}}$) instead of X . Then the second to last restructuring property in Definition 5 would become

$$X_{\{1, \dots, k\}}[X_1(X'_1) \oplus \dots \oplus X_k(X'_k)] \geq X_{\{1, \dots, \ell\}}[X_1(X'_1) \oplus \dots \oplus X_\ell(X'_\ell)].$$

However, as long as we are not dealing with subattributes of the form $X_{\{1, \dots, k\}}[\lambda]$, the additional index does not add any information and thus can be omitted to increase readability. In the last restructuring property in Definition 5, however, the index is needed.

Further note that due to the restructuring rules in Definitions 4 and 5 we may have the case that a record attribute is a subattribute of a list attribute. This allows us to assume that the union-constructor only appears inside a list-constructor or as the outermost constructor. This will be frequently exploited in our proofs.

We use the notation $\mathcal{S}(X) = \{Z \in \mathcal{N} \mid X \geq Z\}$ to denote the *set of subattributes* of a nested attribute X . In the next subsection we will take a closer look into the structure of $\mathcal{S}(X)$.

Figure 1 shows the subattributes of $X[X_1(A) \oplus X_2(B)]$ together with the relation \geq on them. Note that the subattribute $X[\lambda]$ would not occur, if we only considered the record-structure, whereas other subattributes such as $X(X_1[\lambda])$ would not occur, if we only considered the list-structure. This is a direct consequence of the restructuring rules.

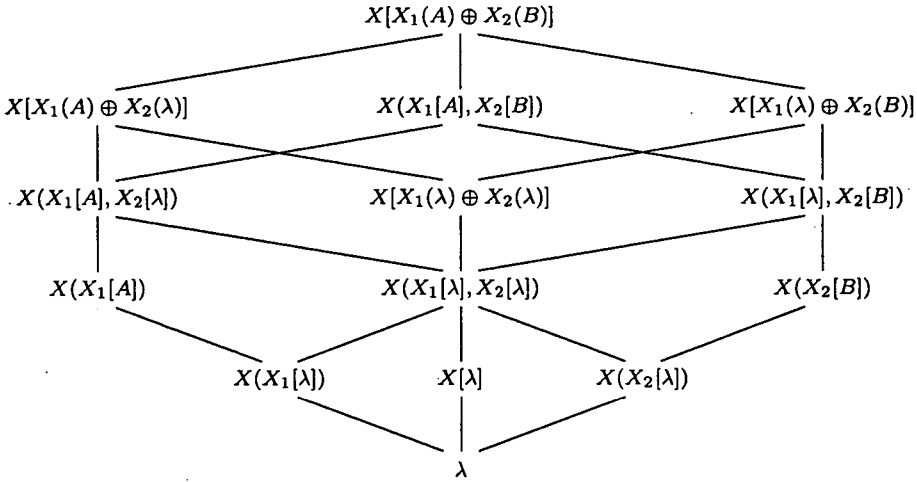
Let us now investigate the structure of $\mathcal{S}(X)$. We will show that we obtain a non-distributive Brouwer algebra, i.e. a non-distributive lattice with relative pseudo-complements. A lattice \mathcal{L} with zero and one, partial order \leq , join \sqcup and meet \sqcap is said to have *relative pseudo-complements* iff for all $Y, Z \in \mathcal{L}$ the infimum $Y \leftarrow Z = \sqcap \{U \mid U \sqcup Y \geq Z\}$ exists.

Proposition 1. *The set $\mathcal{S}(X)$ of subattributes carries the structure of a lattice with zero and one and relative pseudo-complements, where the order \geq is as defined in Definition 5, and λ and X are the zero and one.*

In the following we denote join by \sqcup , meet by \sqcap and relative pseudo-complement by \leftarrow . Then it is straightforward to show the following properties:

- for the join \sqcup :

1. $Y \sqcup Z = Y$ iff $Y \geq Z$;

Figure 1: The lattice $S(X[X_1(A) \oplus X_2(B)])$

2. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ we have $Y \sqcup Z = X(Y_1 \sqcup Z_1, \dots, Y_n \sqcup Z_n)$;
3. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n) \neq \lambda$ and $Z = X_I[\lambda]$ with $I = \{i_1, \dots, i_k\}$ we have $Y \sqcup Z = Z \sqcup Y = Y \sqcup X(X_{i_1}[\lambda], \dots, X_{i_k}[\lambda])$;
4. for $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$, $Y = X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)$ and $Z = X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)$ we have $Y \sqcup Z = X_1(Y'_1 \sqcup Z'_1) \oplus \dots \oplus X_n(Y'_n \sqcup Z'_n)$;
5. for $X = X[X']$, $Y = X[Y']$ and $Z = X[Z']$ we have $Y \sqcup Z = X[Y' \sqcup Z']$;
6. for $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$, $Y = X[X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)]$ and $Z = X(Z_1, \dots, Z_n)$ with either $Z_i = X_i[Z'_i]$ or $Z_i = \lambda$ we have $Y \sqcup Z = Z \sqcup Y = X[X_1(U_1) \oplus \dots \oplus X_n(U_n)]$ with $U_i = \begin{cases} Y'_i \sqcup Z'_i & \text{for } Z_i = X_i[Z'_i] \\ Y'_i & \text{for } Z_i = \lambda \end{cases}$.

• for the meet \sqcap :

1. $Y \sqcap Z = Z$ iff $Y \geq Z$;
2. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ we have $Y \sqcap Z = X(Y_1 \sqcap Z_1, \dots, Y_n \sqcap Z_n)$;
3. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n) \neq \lambda$ and $Z = X_I[\lambda]$ with $I = \{i_1, \dots, i_k\}$ and $Y \not\geq Z$ we have $Y \sqcap Z = Z \sqcap Y = \lambda$;
4. for $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$, $Y = X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)$ and $Z = X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)$ we have $Y \sqcap Z = X_1(Y'_1 \sqcap Z'_1) \oplus \dots \oplus X_n(Y'_n \sqcap Z'_n)$;
5. for $X = X[X']$, $Y = X[Y']$ and $Z = X[Z']$ we have $Y \sqcap Z = X[Y' \sqcap Z']$;

6. for $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$, $Y = X[X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)]$ and $Z = X(Z_1, \dots, Z_n)$ with either $Z_i = X_i[Z'_i]$ or $Z_i = \lambda$ we have $Y \sqcap Z = Z \sqcap Y = X(U_1, \dots, U_n)$ with $U_i = \begin{cases} X_i[Y'_i \sqcap Z'_i] & \text{for } Z_i = X_i[Z'_i] \\ \lambda & \text{for } Z_i = \lambda \end{cases}$.

- for the relative pseudo-complement \leftarrow :

1. $\lambda \leftarrow Y = Y$;
2. for $Y \geq Z$ we have $Y \leftarrow Z = \lambda$;
3. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$, $Z = X(Z_1, \dots, Z_n)$ and $X[\lambda] \notin \mathcal{S}(X)$ we have $Y \leftarrow Z = X(Y_1 \leftarrow Z_1, \dots, Y_n \leftarrow Z_n)$;
4. for $Z = X(Z_1, \dots, Z_n) \neq \lambda$ and $I = \{i_1, \dots, i_k\}$ we have $Z \leftarrow X_I[\lambda] = \lambda$ and $X_I[\lambda] \leftarrow Z = X(X_{i_1}[\lambda] \leftarrow Z_{i_1}, \dots, X_{i_k}[\lambda] \leftarrow Z_{i_k})$;
5. for $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$, $Y = X_1(Y_1) \oplus \dots \oplus X_n(Y_n)$, $Z = X_1(Z_1) \oplus \dots \oplus X_n(Z_n)$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X_1(Y_1 \leftarrow Z_1) \oplus \dots \oplus X_n(Y_n \leftarrow Z_n)$;
6. for $Z = X(Z_1, \dots, Z_n) \neq \lambda$ we have $Z \leftarrow X[\lambda] = \lambda$ and $X[\lambda] \leftarrow Z = X(X_1[\lambda] \leftarrow Z_1, \dots, X_n[\lambda] \leftarrow Z_n)$;
7. for $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$ or $X = X(X_1[X'_1], \dots, X_n[X'_n])$ we have:
 - (a) for $Z = X(Z_1, \dots, Z_n) \neq \lambda$ and $I = \{i_1, \dots, i_k\}$ we have $Z \leftarrow X_I[\lambda] = \lambda$ and $X_I[\lambda] \leftarrow Z = X(X_{i_1}[\lambda] \leftarrow Z_{i_1}, \dots, X_{i_k}[\lambda] \leftarrow Z_{i_k})$;
 - (b) for $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ with $\lambda \neq Y \not\geq Z$
 - if $Y_i \geq Z_i$ or $Y_i = \lambda$, $Z_i = X_i[\lambda]$ for all $i = 1, \dots, n$ we have $Y \leftarrow Z = \lambda$,
 - otherwise we have $Y \leftarrow Z = X(Y_1 \leftarrow Z_1, \dots, Y_n \leftarrow Z_n)$;
 - (c) for $Y = X[X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)]$ and $Z = X(Z_1, \dots, Z_n)$ with $Z_i = X_i[Z'_i]$ or $Z_i = \lambda$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X(U_1, \dots, U_n)$ with $U_i = \begin{cases} X_i[Y'_i \leftarrow Z'_i] & \text{for } Z_i \neq \lambda \\ \lambda & \text{else} \end{cases}$;
 - (d) for $Y = X(Y_1, \dots, Y_n) \neq \lambda$ with $Y_i = X_i[Y'_i]$ or $Y_i = \lambda$, and $Z = X[X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)]$ with $Z_i = X_i(Z'_i)$ or $Z_i = \lambda$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X[X_1(U_1) \oplus \dots \oplus X_n(U_n)]$ with $U_i = \begin{cases} Y'_i \leftarrow Z'_i & \text{for } Y_i \neq \lambda \neq Z_i \\ \lambda & \text{else} \end{cases}$;
 - (e) for $Y = X[X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)]$, $Z = X[X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)]$ with $Y \neq Z$ we have $Y \leftarrow Z = X(U_1, \dots, U_n)$ with

$$U_i = \begin{cases} X_i[Y'_i \leftarrow Z'_i] & \text{for } Y'_i \neq \lambda \neq Z'_i \\ \lambda & \text{else} \end{cases}$$

3 Axiomatisation of Functional Dependencies

In this section we will define functional dependencies on $\mathcal{S}(X)$ and derive some sound derivation rules. We consider finite sets $r \subseteq \text{dom}(X)$, which we will call simply *instances* of X . If Y is a nested attribute that occurs inside X , then an instance r of X defines an instance $r(Y)$ of Y ; simply take $r(Y) = \{v' \in \text{dom}(Y) \mid v' \text{ occurs inside some } v \in r \text{ at the position defined by } Y\}$.

Definition 6. Let $X, X' \in \mathcal{N}$ such that X' occurs in X . A *functional dependency* (FD) on $\mathcal{S}(X)$ is an expression $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ with $\mathcal{Y}, \mathcal{Z} \subseteq \mathcal{S}(X')$.

An instance r of X *satisfies the FD* $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ on $\mathcal{S}(X)$ (notation: $r \models X' : \mathcal{Y} \rightarrow \mathcal{Z}$) iff for all $t_1, t_2 \in r(X')$ with $\pi_Y^{X'}(t_1) = \pi_Y^{X'}(t_2)$ for all $Y \in \mathcal{Y}$ we also have $\pi_Z^{X'}(t_1) = \pi_Z^{X'}(t_2)$ for all $Z \in \mathcal{Z}$.

Let Σ be a set of FDs defined on some $\mathcal{S}(X)$. A FD ψ is implied by Σ (notation: $\Sigma \models \psi$) iff all instances r with $r \models \varphi$ for all $\varphi \in \Sigma$ also satisfy ψ . As usual we write $\Sigma^* = \{\psi \mid \Sigma \models \psi\}$.

We write Σ^+ for the set of all FDs that can be derived from Σ by applying a system \mathfrak{A} of axioms and rules, i.e. $\Sigma^+ = \{\psi \mid \Sigma \vdash_{\mathfrak{A}} \psi\}$. We omit the standard definitions of derivations with a given rule system, and also write simply \vdash instead of $\vdash_{\mathfrak{A}}$, if the rule system is clear from the context.

Our goal is to find a finite axiomatisation, i.e. a rule system \mathfrak{A} such that $\Sigma^* = \Sigma^+$ holds. The rules in \mathfrak{A} are *sound* iff $\Sigma^+ \subseteq \Sigma^*$ holds, and *complete* iff $\Sigma^* \subseteq \Sigma^+$ holds.

3.1 Sound Axioms and Rules for Functional Dependencies

Let us now look at derivation rules for FDs. We will need a particular notion of “semi-disjointness” that will permit a generalisation of the well known Armstrong axioms for the relational model.

Definition 7. Two subattributes $Y, Z \in \mathcal{S}(X)$ are called *semi-disjoint* iff one of the following holds:

1. $Y \geq Z$ or $Z \geq Y$;
2. $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$, $Z = X(Z_1, \dots, Z_n)$ and $Y_i, Z_i \in \mathcal{S}(X_i)$ are semi-disjoint for all $i = 1, \dots, n$;
3. $X = X[X']$, $Y = X[Y']$, $Z = X[Z']$ and $Y', Z' \in \mathcal{S}(X')$ are semi-disjoint;
4. $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$, $Y = X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)$, $Z = X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)$ and $Y'_i, Z'_i \in \mathcal{S}(X'_i)$ are semi-disjoint for all $i = 1, \dots, n$;
5. $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$, $Y = X(Y_1, \dots, Y_n)$ with $Y_i = X_i[Y'_i]$ or $Y_i = \lambda = Y'_i$, $Z = X[X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)]$, and Y'_i, Z'_i are semi-disjoint for all $i = 1, \dots, n$.

With the notion of semi-disjointness we can formulate axioms and rules for FDs and show their soundness.

Theorem 1. *The following axioms and rules are sound for the implication of FDs:*

- the λ axiom: $X' : \emptyset \rightarrow \{\lambda\}$
- the subattribute axiom: $X' : \{Y\} \rightarrow \{Z\}$ for $Y \geq Z$
- the join axiom: $X' : \{Y, Z\} \rightarrow \{Y \sqcup Z\}$ for semi-disjoint Y and Z
- the reflexivity axiom: $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ for $\mathcal{Z} \subseteq \mathcal{Y}$
- the extension rule: $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ implies $X' : \mathcal{Y} \rightarrow \mathcal{Y} \cup \mathcal{Z}$
- the transitivity rule: $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ and $X' : \mathcal{Z} \rightarrow \mathcal{U}$ imply $X' : \mathcal{Y} \rightarrow \mathcal{U}$
- the list axioms:
 - $X : \{X_I[\lambda], X_J[\lambda]\} \rightarrow \{X_{I \cup J}[\lambda]\}$ for $I \cap J = \emptyset$
 - $X : \{X_I[\lambda], X_{I \cup J}[\lambda]\} \rightarrow \{X_J[\lambda]\}$ for $I \cap J = \emptyset$
 - $X : \{X_I[\lambda], X_J[\lambda], X_{I \cap J}[\lambda]\} \rightarrow \{X_{(I-J) \cup (J-I)}[\lambda]\}$
 - $X : \{X_I[\lambda], X_J[\lambda], X_{(I-J) \cup (J-I)}[\lambda]\} \rightarrow \{X_{I \cap J}[\lambda]\}$
- the list lifting rule: $X' : \mathcal{Y} \rightarrow \mathcal{Z}$ implies $X[X'] : \{X[Y] \mid Y \in \mathcal{Y}\} \rightarrow \{X[Z] \mid Z \in \mathcal{Z}\}$ for $\mathcal{Y} \neq \emptyset$
- the record lifting rule: $X_i : \mathcal{Y}_i \rightarrow \mathcal{Z}_i$ implies $X(X_1, \dots, X_n) : \bar{\mathcal{Y}}_i \rightarrow \bar{\mathcal{Z}}_i$ with $\bar{\mathcal{Y}}_i = \{X(\lambda, \dots, Y_i, \dots, \lambda) \mid Y_i \in \mathcal{Y}_i\}$ and $\bar{\mathcal{Z}}_i = \{X(\lambda, \dots, Z_i, \dots, \lambda) \mid Z_i \in \mathcal{Z}_i\}$
- the union lifting rule: $X'_i : \mathcal{Y}_i \rightarrow \mathcal{Z}_i$ implies $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) : \bar{\mathcal{Y}}_i \rightarrow \bar{\mathcal{Z}}_i$ with $\bar{\mathcal{Y}}_i = \{X_1(\lambda) \oplus \dots \oplus X_i(Y_i) \oplus \dots \oplus X_n(\lambda) \mid Y_i \in \mathcal{Y}_i\}$ and $\bar{\mathcal{Z}}_i = \{X_1(\lambda) \oplus \dots \oplus X_i(Z_i) \oplus \dots \oplus X_n(\lambda) \mid Z_i \in \mathcal{Z}_i\}$

Proof. We only show the soundness of some of the axioms and rules. The proof for the other axioms and rules is either analogous or trivial.

For the join axiom let $t_1, t_2 \in \text{dom}(X)$ with $\pi_Y^X(t_1) = \pi_Y^X(t_2)$ and $\pi_Z^X(t_1) = \pi_Z^X(t_2)$. We use induction on X to show $\pi_{Y \sqcup Z}^X(t_1) = \pi_{Y \sqcup Z}^X(t_2)$. The cases $X = \lambda$ and $X = A$ (i.e. a simple attribute) are trivial. There is also nothing to show for $Y \geq Z$ or $Z \geq Y$, as in these cases $Y \sqcup Z$ is one of Y or Z .

For $X = X(X_1, \dots, X_n)$ let $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ be semi-disjoint. For $t_j = (X_1 : t_{j1}, \dots, X_n : t_{jn})$ ($j = 1, 2$) we have $\pi_{Y_i}^{X_i}(t_{1i}) = \pi_{Y_i}^{X_i}(t_{2i})$ and $\pi_{Z_i}^{X_i}(t_{1i}) = \pi_{Z_i}^{X_i}(t_{2i})$, and Y_i, Z_i are semi-disjoint for all $i = 1, \dots, n$. By induction $\pi_{Y_i \sqcup Z_i}^{X_i}(t_{1i}) = \pi_{Y_i \sqcup Z_i}^{X_i}(t_{2i})$, which implies $\pi_Y^X(t_1) = \pi_Y^X(t_2)$.

For $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$ assume $t_1 = (X_j : t'_1)$ and $t_2 = (X_j : t'_2)$. Thus, for semi-disjoint $Y = X_1(Y'_1) \oplus \dots \oplus X_n(Y'_n)$ and $Z = X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)$

we obtain $\pi_{Y'_j}^{X'_j}(t'_1) = \pi_{Y'_j}^{X'_j}(t'_2)$, $\pi_{Z'_j}^{X'_j}(t'_1) = \pi_{Z'_j}^{X'_j}(t'_2)$, and Y'_j, Z'_j are semi-disjoint. By induction $\pi_{Y'_j \sqcup Z'_j}^{X'_j}(t'_1) = \pi_{Y'_j \sqcup Z'_j}^{X'_j}(t'_2)$, which implies $\pi_{Y \sqcup Z}^X(t_1) = \pi_{Y \sqcup Z}^X(t_2)$.

For $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$, $Y = X(Y_1, \dots, Y_n)$ with $Y_i = X_i[Y'_i]$ or $Y_i = \lambda = Y'_i$ and $Z = X[X_1(Z'_1) \oplus \dots \oplus X_n(Z'_n)]$ we get $Y \sqcup Z = X[X_1(Y'_1 \sqcup Z'_1) \oplus \dots \oplus X_n(Y'_n \sqcup Z'_n)]$. As $Z \geq X[\lambda]$, we also have $\pi_{X[\lambda]}^X(t_1) = \pi_{X[\lambda]}^X(t_2)$, so t_1 and t_2 are lists of equal length. Therefore, assume $t_j = [t_{j1}, \dots, t_{jm}]$ for $j = 1, 2$ and $t_{jk} = (X_\ell : t''_{jk})$. This gives $\pi_{Y \sqcup Z}^X(t_j) = [t'_{j1}, \dots, t'_{jm}]$ with $t'_{jk} = (X_\ell : \pi_{Y'_\ell \sqcup Z'_\ell}^{X'_\ell}(t''_{jk}))$. We know $\pi_{Z'_\ell}^{X'_\ell}(t''_{1k}) = \pi_{Z'_\ell}^{X'_\ell}(t''_{2k})$, so we are done for $Y_\ell = \lambda$. For $Y_\ell \neq \lambda$ the sublists containing all $(X_\ell : t'_{jk})$ coincide on Y'_ℓ . As Y'_ℓ and Z'_ℓ are semi-disjoint, we have $\pi_{Y'_\ell \sqcup Z'_\ell}^{X'_\ell}(t''_{1k}) = \pi_{Y'_\ell \sqcup Z'_\ell}^{X'_\ell}(t''_{2k})$ by induction, which implies $\pi_{Y \sqcup Z}^X(t_1) = \pi_{Y \sqcup Z}^X(t_2)$.

For the first list axiom let $t_1, t_2 \in \text{dom}(X)$. Then $\pi_{X_I[\lambda]}^X(t_1) = \pi_{X_I[\lambda]}^X(t_2)$ means that t_1 and t_2 contain the same number of elements of the form $(X_i : v_i)$ with $i \in I$. If the same holds for $I \cup J$, then t_1 and t_2 must also contain the same number of elements of the form $(X_i : v_i)$ with $i \in J$, i.e. $\pi_{X_J[\lambda]}^X(t_1) = \pi_{X_J[\lambda]}^X(t_2)$. The soundness of the second list axiom follows from the same argument.

Analogously, for the third list axiom $\pi_Y^X(t_1) = \pi_Y^X(t_2)$ for $Y \in \{X_I[\lambda], X_J[\lambda], X_{I \cap J}[\lambda]\}$ means that t_1, t_2 contain the same number of elements with labels in I, J and $I \cap J$, respectively. So they also contain the same number of elements with labels in $(I - J) \cup (J - I)$. The soundness of the fourth list axiom follows from the same argument.

For the soundness of the list lifting rule let $t_1, t_2 \in \text{dom}(X)$ with $\pi_{X[Y]}^X(t_1) = \pi_{X[Y]}^X(t_2)$ for all $X[Y]$ with $Y \in \mathcal{Y}$. As $\mathcal{Y} \neq \emptyset$, it follows that t_1 and t_2 must have the same length, say $t_i = [t_{i1}, \dots, t_{ik}]$ ($i = 1, 2$), and for all $j = 1, \dots, k$ and all $Y \in \mathcal{Y}$ we have $\pi_Y^{X'}(t_{1j}) = \pi_Y^{X'}(t_{2j})$. Hence $\pi_Z^{X'}(t_{1j}) = \pi_Z^{X'}(t_{2j})$ for all $j = 1, \dots, k$ and all $Z \in \mathcal{Z}$, which implies $\pi_{X[Z]}^X(t_1) = \pi_{X[Z]}^X(t_2)$ for all $X[Z]$ with $Z \in \mathcal{Z}$. The soundness of the other two lifting rules follows analogously. \square

Using these rules we can derive additional rules:

- the union rule: $X : \mathcal{Y} \rightarrow \mathcal{Z}$ and $X : \mathcal{Y} \rightarrow \mathcal{U}$ imply $X : \mathcal{Y} \rightarrow \mathcal{Z} \cup \mathcal{U}$
- the fragmentation rule: $X : \mathcal{Y} \rightarrow \mathcal{Z}$ implies $X : \mathcal{Y} \rightarrow \{Z\}$ for $Z \in \mathcal{Z}$
- the join rule: $X : \{Y\} \rightarrow \{Z\}$ implies $X : \{Y\} \rightarrow \{Y \sqcup Z\}$ for semi-disjoint Y and Z

3.2 SHL-Ideals

In this subsection we investigate ideals. Of particular interest will be ideals with additional closure properties, which we call “strong high-level ideals” or SHL-ideals for short. These ideals will appear naturally in the completeness proof in the next subsection. The main result of this subsection is Theorem 2.

Definition 8. An *ideal* for a nested attribute X is a subset $\mathcal{G} \subseteq \mathcal{S}(X)$ with $\lambda \in \mathcal{G}$ and whenever $Y \in \mathcal{G}$, $Z \in \mathcal{S}(X)$ with $Y \geq Z$, then also $Z \in \mathcal{G}$.

Let us now address the closure properties that will turn ideals into “higher-level” or “strong higher-level ideals”.

Definition 9. Let $X \in \mathcal{N}$. An ideal $\mathcal{F} \subseteq \mathcal{S}(X)$ is called *SHL-ideal* on $\mathcal{S}(X)$ iff the following properties are satisfied:

1. if $Y, Z \in \mathcal{F}$ are semi-disjoint, then $Y \sqcup Z \in \mathcal{F}$;
2. (a) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$ with $I \subseteq J$, then $X_{J-I}[\lambda] \in \mathcal{F}$;
 (b) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$ with $I \cap J = \emptyset$, then $X_{I \cup J}[\lambda] \in \mathcal{F}$;
 (c) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$, then $X_{I \cap J}[\lambda] \in \mathcal{F}$ iff $X_{(I-J) \cup (J-I)}[\lambda] \in \mathcal{F}$;
3. if $X = X(X'_1, \dots, X'_n)$, then the sets $\mathcal{F}_i = \{Y_i \in \mathcal{S}(X'_i) \mid X(\lambda, \dots, Y_i, \dots, \lambda) \in \mathcal{F}\}$ are SHL-ideals;
4. if $X = X[X']$ and $\mathcal{F} \neq \{\lambda\}$, then the set $\mathcal{G} = \{Y \in \mathcal{S}(X') \mid X[Y] \in \mathcal{F}\}$ is a SHL-ideal;
5. If $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$ and $\mathcal{F} \neq \{\lambda\}$, then the sets $\mathcal{F}_i = \{Y_i \in \mathcal{S}(X'_i) \mid X_1(\lambda) \oplus \dots \oplus X_i(Y_i) \oplus \dots \oplus X_n(\lambda) \in \mathcal{F}\}$ are SHL-ideals.

We now prove the main result of this subsection.

Theorem 2. Let X be a nested attribute such that the union-constructor appears in X only inside a list-constructor. If \mathcal{F} is a SHL-ideal on $\mathcal{S}(X)$, then there exist tuples $t_0, t_1 \in \text{dom}(X)$ with $\pi_Y^X(t_0) = \pi_Y^X(t_1)$ iff $Y \in \mathcal{F}$.

Proof. We use induction on X . The case $X = \lambda$ is trivial. For a simple attribute $X = A$ we either have $\mathcal{F} = \{\lambda\}$ or $\mathcal{F} = \{A, \lambda\}$. In the former case take $t_0 = a$ and $t_1 = a'$ for $a, a' \in \text{dom}(A)$ with $a \neq a'$. In the latter case take $t_0 = t_1 = a$.

Let $X = X(X_1, \dots, X_n)$ with $X[\lambda] \notin \mathcal{S}(X)$. Take the SHL-ideals \mathcal{F}_i from Definition 9(3). By induction we find $t_{0i}, t_{1i} \in \text{dom}(X_i)$ with $\pi_{Y_i}^{X_i}(t_{0i}) = \pi_{Y_i}^{X_i}(t_{1i})$ iff $Y_i \in \mathcal{F}_i$. So take $t_j = (X_1 : t_{j1}, \dots, X_n : t_{jn})$ ($j = 0, 1$). For $Y = X(Y_1, \dots, Y_n) \in \mathcal{F}$ we have $\pi_Y^X(t_0) = (X_1 : \pi_{Y_1}^{X_1}(t_{01}), \dots, X_n : \pi_{Y_n}^{X_n}(t_{0n})) = (X_1 : \pi_{Y_1}^{X_1}(t_{11}), \dots, X_n : \pi_{Y_n}^{X_n}(t_{1n})) = \pi_Y^X(t_1)$. For $Y = X(Y_1, \dots, Y_n) \notin \mathcal{F}$ there is at least one $Y_i \notin \mathcal{F}_i$, which gives $\pi_Y^X(t_0) = (X_1 : \pi_{Y_1}^{X_1}(t_{01}), \dots, X_n : \pi_{Y_n}^{X_n}(t_{0n})) \neq (X_1 : \pi_{Y_1}^{X_1}(t_{11}), \dots, X_n : \pi_{Y_n}^{X_n}(t_{1n})) = \pi_Y^X(t_1)$.

Let $X = X[X']$ and assume that X' is not a union attribute. If we have $\mathcal{F} = \{\lambda\}$, then take $t_0 = [v]$ with $v \in \text{dom}(X')$ and $t_1 = []$. For $Y = X[Y'] \notin \mathcal{F}$ we get $\pi_Y^X(t_0) = [\pi_{Y'}^{X'}(v)] \neq [] = \pi_Y^X(t_1)$. For $\mathcal{F} \neq \{\lambda\}$ take the SHL-ideal \mathcal{G} from Definition 9(4). By induction we find $t'_0, t'_1 \in \text{dom}(X')$ with $\pi_{Y'}^{X'}(t'_0) = \pi_{Y'}^{X'}(t'_1)$ iff $Y' \in \mathcal{G}$. Let $t_j = [t'_j]$ for $j = 0, 1$. Then we get $\pi_Y^X(t_0) = [\pi_{Y'}^{X'}(t'_0)] = [\pi_{Y'}^{X'}(t'_1)] = \pi_Y^X(t_1)$ iff $Y = X[Y'] \in \mathcal{F}$.

Let $X = X[X_1(X'_1) \oplus \dots \oplus X_n(X'_n)]$. If $\mathcal{F} = \{\lambda\}$, define $t_0 = [(X_1 : v_1), \dots, (X_n : v_n)]$ with arbitrary $v_i \in \text{dom}(X'_i)$ and $t_1 = []$. Then we get $\pi_{X_I[\lambda]}^X(t_0) = \underbrace{[\top, \dots, \top]}_{|I| \text{ times}}$,

whereas $\pi_{X_I[\lambda]}^X(t_1) = []$.

Now assume $\mathcal{F} \neq \{\lambda\}$. Take $I^+ = \{i \in \{1, \dots, n\} \mid X(X_i[\lambda]) \in \mathcal{F}\}$ and $I^- = \{1, \dots, n\} - I^+$. If $I^+ = \{i_1, \dots, i_k\}$, then consider first the subattribute $X^+ = X[X_{i_1}(X'_{i_1}) \oplus \dots \oplus X_{i_k}(X'_{i_k})]$. By Definition 9(2b) we have $X_I[\lambda] \in \mathcal{F}$ for all $I \subseteq I^+$. We first construct $t_0^+, t_1^+ \in \text{dom}(X^+)$ with $\pi_Y^{X^+}(t_0^+) = \pi_Y^{X^+}(t_1^+)$ iff $Y \in \mathcal{F}^+ = \{Y \in \mathcal{F} \mid X^+ \geq Y\}$.

For this take $\tilde{X} = X(X_{i_1}[X'_{i_1}], \dots, X_{i_k}[X'_{i_k}])$ and $\mathcal{H} = \{Y = X(Y_{i_1}, \dots, Y_{i_k}) \mid Y \in \mathcal{F}\}$. Ignoring restructuring and considering \tilde{X} just as a record attribute, \mathcal{H} becomes an SHL-ideal on $\mathcal{S}(\tilde{X})$. Applying the record case above we obtain $\tilde{t}_0, \tilde{t}_1 \in \text{dom}(\tilde{X})$ with $\pi_Y^{\tilde{X}}(\tilde{t}_0) = \pi_Y^{\tilde{X}}(\tilde{t}_1)$ iff $Y \in \mathcal{H}$.

If $\tilde{t}_i = (t_{i,i_1}, \dots, t_{i,i_k})$, we may concatenate these lists in the order of the indices to define t_0^+ and t_1^+ , respectively. Then for $Y \in \mathcal{S}(X^+)$ with $\tilde{X} \geq Y$ we have $\pi_Y^{X^+}(t_0^+) = \pi_Y^{X^+}(t_1^+)$ iff $Y \in \mathcal{F}^+$. This does not change, if for any j we replace t_{0,i_j} and t_{1,i_j} by the concatenated lists $t_{0,i_j} \frown t_{0,i_j}$ and $t_{1,i_j} \frown t_{0,i_j}$, respectively.

Now let $K = \{k_1, \dots, k_m\} \subseteq I^+$ be maximal such that $X[X_{k_1}(X'_{k_1}) \oplus \dots \oplus X_{k_m}(X'_{k_m})] \in \mathcal{F}$. Then for $k \in I^+ - K$ we must have $X(X_k[X'_k]) \notin \mathcal{F}$, otherwise also $X[X_{k_1}(X'_{k_1}) \oplus \dots \oplus X_{k_m}(X'_{k_m}) \oplus X_k(X'_k)] \in \mathcal{F}$ due to the semi-disjointness of the two subattributes and property 1 in Definition 9. Therefore, K is uniquely determined.

Now, if $X(X_{i_1}[Y'_{i_1}], \dots, X_{i_\mu}[Y'_{i_\mu}]) \in \mathcal{F}$, but $X[X_{i_1}(Y'_{i_1}) \oplus \dots \oplus X_{i_\mu}(Y'_{i_\mu})] \notin \mathcal{F}$, then the uniqueness of K implies $X(X_{i_1}[X'_{i_1}], \dots, X_{i_\mu}[X'_{i_\mu}]) \notin \mathcal{F}$. Hence there must be some $\iota \in \{i_1, \dots, i_\mu\}$ with $t_{0,\iota} \neq t_{1,\iota}$. We therefore replace $t_{0,\iota}$ and $t_{1,\iota}$ by the concatenated lists $t_{0,\iota} \frown t_{0,\iota}$ and $t_{1,\iota} \frown t_{0,\iota}$, respectively, changing t_0^+ and t_1^+ accordingly. This gives $\pi_{X[X_{i_1}(Y'_{i_1}) \oplus \dots \oplus X_{i_\mu}(Y'_{i_\mu})]}^{X^+}(t_0^+) \neq \pi_{X[X_{i_1}(Y'_{i_1}) \oplus \dots \oplus X_{i_\mu}(Y'_{i_\mu})]}^{X^+}(t_1^+)$ without destroying previously established equalities and inequalities. This implies $\pi_Y^{X^+}(t_0^+) = \pi_Y^{X^+}(t_1^+)$ iff $Y \in \mathcal{F}^+$ for all $Y \in \mathcal{S}(X^+)$ as claimed.

Now let $I^- = \{j_1, \dots, j_\ell\}$. We choose non-negative integers x_i, y_i ($i = 1, \dots, \ell$) such that for each $I = \{j_{r_1}, \dots, j_{r_{|I|}}\} \subseteq I^-$ we have

$$\sum_{p=1}^{|I|} x_{r_p} = \sum_{p=1}^{|I|} y_{r_p} \text{ iff } X_I[\lambda] \in \mathcal{F}.$$

These integers can be obtained by the following procedure:

```

for  $p = 1, \dots, \ell$  :
  choose  $x_p, y_p$  such that all equations and inequations containing
  only  $x_i, y_i$  with  $1 \leq i \leq p$  are satisfied;
  replace  $x_p, y_p$  in the remaining equations and inequations by the
  chosen values
endfor

```

Properties 2(b) and 2(c) in Definition 9 guarantee that this procedure always produces a solution for the given equations and inequations. Now define

$$t_0^- = [\underbrace{(X_{j_1} : v_{j_1})}_{x_{j_1}\text{-times}}, \dots, \underbrace{(X_{j_\ell} : v_{j_\ell})}_{x_{j_\ell}\text{-times}}] \quad \text{and} \quad t_1^- = [\underbrace{(X_{j_1} : v_{j_1})}_{y_{j_1}\text{-times}}, \dots, \underbrace{(X_{j_\ell} : v_{j_\ell})}_{y_{j_\ell}\text{-times}}]$$

with arbitrary values $v_{j_i} \in \text{dom}(X'_{j_i})$ and concatenate t_i^+ and t_i^- to give t_i ($i = 0, 1$). Then for $Y \in \{Y \in \mathcal{S}(X) \mid X^+ \geq Y\}$ we have $\pi_Y^X(t_i) = \pi_Y^X(t_i^+)$, hence $\pi_Y^X(t_0) = \pi_Y^X(t_1)$ iff $Y \in \mathcal{F}^+$.

For $Y \not\leq X^+$ we always have one $j \in I^-$ with $Y \geq X(X_j[\lambda])$ or $Y = X_I[\lambda]$. In the first case $Y \notin \mathcal{F}$ and

$$\pi_{X(X_j[\lambda])}^X(t_0) = \pi_{X(X_j[\lambda])}^X(t_0^-) \neq \pi_{X(X_j[\lambda])}^X(t_1^-) = \pi_{X(X_j[\lambda])}^X(t_1)$$

as desired. In the second case $\pi_Y^X(t_0) = \pi_Y^X(t_1)$ iff $\pi_{X_{I \cap I^-}[\lambda]}^X(t_0^-) = \pi_{X_{I \cap I^-}[\lambda]}^X(t_1^-)$ iff $X_{I \cap I^-}[\lambda] \in \mathcal{F}$ iff $Y = X_I[\lambda] \in \mathcal{F}$ due to property 2(a) of Definition 9 and $X_{I \cap I^+}[\lambda] \in \mathcal{F}$. \square

3.3 Completeness of the Axioms and Rules for Functional Dependencies

In this final subsection we want to show that the axioms and rules from Theorem 1 are also complete. This gives our main result.

Before we come to the proof let us make a little observation on the union-constructor. If $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$, then each instance r of X can be partitioned into r_i ($i = 1, \dots, n$), where r_i contains exactly the X_i -labelled elements of r . Then r satisfies a FD $\varphi \equiv \mathcal{Y} \rightarrow \mathcal{Z}$ iff each r_i satisfies the i 'th projection φ_i of φ , which results by replacing all subattributes $Y = X_1(Y_1) \oplus \dots \oplus X_n(Y_n)$ in \mathcal{Y} or \mathcal{Z} by $X_i(Y_i)$. Similarly, we see $\varphi \in \Sigma^+$ iff $\varphi_i \in \Sigma_i^+$ for all $i = 1, \dots, n$.

Theorem 3. *The axioms and rules in Theorem 1 are complete for the implication of FDs.*

Proof. Assume $\mathcal{Y} \rightarrow \mathcal{Z} \notin \Sigma^+$. Due to the union rule we must have $\mathcal{Y} \rightarrow \{Z\} \notin \Sigma^+$ for some $Z \in \mathcal{Z}$. Now take $\bar{\mathcal{Y}} = \{Z \mid \mathcal{Y} \rightarrow \{Z\} \in \Sigma^+\}$, so $Z \notin \bar{\mathcal{Y}}$. It is easy to see that $\mathcal{F} = \bar{\mathcal{Y}}$ is a SHL-ideal:

1. $\lambda \in \bar{\mathcal{Y}}$ follows from the reflexivity axiom, the subattribute axiom and the transitivity rule.
2. In the same way for $Z \in \bar{\mathcal{Y}}$ and $Z \geq Z'$ we get $Z' \in \bar{\mathcal{Y}}$ from the subattribute axiom and the transitivity rule.
3. For semi-disjoint $Z, Z' \in \bar{\mathcal{Y}}$ we obtain $Z \sqcup Z' \in \bar{\mathcal{Y}}$ from the union rule, the join axiom and the transitivity rule.

4. The other properties of SHL-ideals follow directly from the list axioms and the lifting rules.

If the outermost constructor is not the union-constructor, we can apply Theorem 2 to obtain an instance $r = \{t_1, t_2\}$ with $\pi_Z^X(t_1) = \pi_Z^X(t_2)$ iff $Z \in \bar{Y}$. As $Y \subseteq \bar{Y}$ and $Z \notin \bar{Y}$, we must have $r \not\models Y \rightarrow \{Z\}$ and thus also $r \not\models Y \rightarrow Z$ due to the soundness of the fragmentation rule.

If the outermost constructor is the union-constructor, say $X = X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$ and thus $Z = X_1(Z_1) \oplus \dots \oplus X_n(Z_n)$, we find some i with $Z_i \notin \mathcal{F}_i$. Otherwise all $X_1(\lambda) \oplus \dots \oplus X_i(Z_i) \oplus \dots \oplus X_n(\lambda) \in \mathcal{F}$, and as these attributes are all semi-disjoint, we would obtain $Z \in \mathcal{F}$, too, which contradicts our assumptions.

Apply the Theorem 2 to X'_i and \mathcal{F}_i , which gives $t_{i1}, t_{i2} \in \text{dom}(X'_i)$ with $\pi_{Y_i}^{X'_i}(t_{i1}) = \pi_{Y_i}^{X'_i}(t_{i2})$ iff $Y_i \in \mathcal{F}_i$. Take $r = \{(X_i : t_{i1}), (X_i : t_{i2})\}$. For $Y \in \mathcal{Y}$ we get

$$\pi_Y^X((X_i : t_{i1})) = (X_i : \pi_{Y_i}^{X'_i}(t_{i1})) = (X_i : \pi_{Y_i}^{X'_i}(t_{i2})) = \pi_Y^X((X_i : t_{i2}))$$

and on the other hand

$$\pi_Z^X((X_i : t_{i1})) = (X_i : \pi_{Z_i}^{X'_i}(t_{i1})) \neq (X_i : \pi_{Z_i}^{X'_i}(t_{i2})) = \pi_Z^X((X_i : t_{i2})),$$

i.e. $r \not\models Y \rightarrow \{Z\}$ and thus also $r \not\models Y \rightarrow Z$ follows also in this case.

In order to complete the proof we have to show $r \models \Sigma$. Let $X' : \mathcal{V} \rightarrow \mathcal{W} \in \Sigma$. Applying only the lifting rules we obtain $X : \mathcal{V}^* \rightarrow \mathcal{W}^* \in \Sigma^+$. We either have $\mathcal{V}^* \subseteq \bar{Y}$ or not. In the first case we obtain $Y \rightarrow \mathcal{V}^* \in \Sigma^+$ and thus also $Y \rightarrow \mathcal{W}^* \in \Sigma^+$, which implies $\mathcal{W}^* \subseteq \bar{Y}$. This gives $\pi_W^X(t_1) = \pi_W^X(t_2)$ for all $W \in \mathcal{W}^*$ and hence $r(X') \models \mathcal{V} \rightarrow \mathcal{W}$. If $\mathcal{V}^* \not\subseteq \bar{Y}$, then there is some $V \in \mathcal{V}^* - \bar{Y}$, for which we must have $\pi_V^X(t_1) \neq \pi_V^X(t_2)$. This implies also $r(X') \models \mathcal{V} \rightarrow \mathcal{W}$. \square

4 Conclusion

In this article we extended our theory of functional dependencies for higher-order data models and presented an axiomatisation for functional dependencies over XML documents. The approach is based on a representation of XML document type definitions (or XML schemata) by nested attributes using constructors for records, disjoint unions and lists, and a particular null value, which covers optionality. The list-constructor is used to represent the Kleene-star in regular expressions in XML element definitions.

In order to fully capture functional dependencies over XML documents we extended our previous work in two major directions. We introduced rational tree attributes, which result from reference structures in XML documents. This led to infinite subattribute lattices, but did not affect the derivation of dependencies. This is the first time that the investigation of functional dependencies for XML did not neglect references. Furthermore, we considered functional dependencies on embedded elements. These dependencies can be lifted through the constructors, i.e. they

induce functional dependencies on complete XML documents. Such dependencies have not been considered in previous work.

Using a partial order on nested attributes we obtain the structure of non-distributive Brouwer algebras. The operations of the Brouwer algebra are exploited in the soundness and completeness proofs for derivation rules for functional dependencies.

In our theory it is also possible to use the multiset- or set-constructor instead of the list-constructor, thus neglecting order or duplicates. We may also treat all three “bulk” constructors together. These extensions had to be left out in this article. A glimpse of the necessary extensions for the other two bulk constructors can be obtained from [10].

Natural next steps in the development of a fully satisfying dependency theory for XML will be the generalisation to other classes of dependencies, e.g. multi-valued or join dependencies, the investigation of efficient closure algorithms, and the study of normal forms [15] that provably characterise desirable properties of well-designed XML documents. First steps in this direction are the normal forms introduced in [3], [8], and [17].

References

- [1] ABITEBOUL, S., BUNEMAN, P., AND SUCIU, D. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.
- [2] ABITEBOUL, S., AND HULL, R. Restructuring hierarchical database objects. *Theoretical Computer Science* (1988).
- [3] ARENAS, M., AND LIBKIN, L. A normal form for XML documents. In *PODS 2002* (2002), ACM.
- [4] BUNEMAN, P., DAVIDSON, S., FAN, W., HARA, C., AND TAN, W. Keys for XML. In *Tenth WWW Conference* (2001), IEEE.
- [5] GOLDFARB, C., AND PRESCOD, P. *The XML Handbook*. Prentice Hall, 1998.
- [6] HARTMANN, S., HOFFMANN, A., LINK, S., AND SCHEWE, K.-D. Axiomatizing functional dependencies in the higher-order entity relationship model. *Information Processing Letters* 87, 3 (2003), 133–137.
- [7] HARTMANN, S., AND LINK, S. Reasoning about functional dependencies in an abstract data model. *Electronic Notes in Theoretical Computer Science* 84 (2003).
- [8] HARTMANN, S., LINK, S., AND SCHEWE, K.-D. A new normal form for conceptual databases. In *Information Modelling and Knowledge Bases XV* (2004), Y. Kiyoki, E. Kawaguchi, H. Jaakkola, and H. Kangassalo, Eds., vol. 105 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 88–105.

- [9] HARTMANN, S., LINK, S., AND SCHEWE, K.-D. Reasoning about functional and multi-valued dependencies in the presence of lists. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.
- [10] HARTMANN, S., LINK, S., AND SCHEWE, K.-D. Weak functional dependencies in higher-order datamodels. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.
- [11] SALI, A. Minimal keys in higher-order datamodels. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.
- [12] SCHEWE, K.-D., AND THALHEIM, B. Fundamental concepts of object oriented databases. *Acta Cybernetica* 11, 4 (1993), 49–85.
- [13] THALHEIM, B. *Dependencies in Relational Databases*. Teubner-Verlag, 1991.
- [14] THALHEIM, B. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag, 2000.
- [15] VINCENT, M. *The semantic justification for normal forms in relational database design*. PhD thesis, Monash University, Melbourne, Australia, 1994.
- [16] VINCENT, M. W., AND LIU, J. Functional dependencies for XML. In *Web Technologies and Applications: 5th Asia-Pacific Web Conference* (2003), vol. 2642 of *LNCS*, Springer-Verlag, pp. 22–34.
- [17] VINCENT, M. W., AND LIU, J. Multivalued dependencies and a 4nf for XML. In *Advanced Information Systems Engineering: 15th International Conference CAiSE 2003* (2003), vol. 2681 of *LNCS*, Springer-Verlag, pp. 14–29.
- [18] VINCENT, M. W., AND LIU, J. Multivalued dependencies in XML. In *British National Conference on Database Systems: BNCOD 2003* (2003), vol. 2712 of *LNCS*, Springer-Verlag, pp. 4–18.

Received May, 2004

Some Results Related to Dense Families of Database Relations

Vu Duc Thi* and Nguyen Hoang Son†

Abstract

The dense families of database relations were introduced by Järvinen [7]. The aim of this paper is to investigate some new properties of dense families of database relations, and their applications. That is, we characterize functional dependencies and minimal keys in terms of dense families. We give a necessary and sufficient condition for an arbitrary family to be R -dense family. We prove that with a given relation R the equality set E_R is an R -dense family whose size is at most $\frac{m(m-1)}{2}$, where m is the number of tuples in R . We also prove that the set of all minimal keys of relation R is the transversal hypergraph of the complement of the equality set E_R . We give an effective algorithm finding all minimal keys of a given relation R . We also give an algorithm which from a given relation R finds a cover of functional dependencies that holds in R . The complexity of these algorithms is also estimated.

1 Basic definitions

In this section we present briefly the main concepts of the theory of relational databases which will be needed in sequel. The concepts and facts given in this section can be found in [1, 3, 4, 8, 9].

Let U be a finite set of *attributes* (e.g. name, age etc). The elements of U will be denoted by a, b, c, \dots, x, y, z , if an ordering on U is needed, by a_1, \dots, a_n . A map dom associates with each $a \in U$ its *domain* $dom(a)$. A *relation* R over U is a subset of Cartesian product $\prod_{a \in U} dom(a)$.

We can think of a relation R over U as being a set of tuples: $R = \{h_1, \dots, h_m\}$,

$$h_i : U \longrightarrow \bigcup_{a \in U} dom(a), h_i(a) \in dom(a), i = 1, 2, \dots, m.$$

A *functional dependency* (FD for short) is a statement of form $X \rightarrow Y$, where $X, Y \subseteq U$. The FD $X \rightarrow Y$ holds in a relation $R = \{h_1, \dots, h_m\}$ over U if

$$(\forall h_i, h_j \in R)((\forall a \in X)(h_i(a) = h_j(a)) \Rightarrow (\forall b \in Y)(h_i(b) = h_j(b))).$$

*Institute of Information Technology, Vietnamese Academy of Science and Technology, 18 Hoang Quoc Viet, Hanoi, Vietnam.

†Department of Mathematics, College of Sciences, Hue University, Vietnam.

We also say that R satisfies the FD $X \rightarrow Y$.

Let F_R be a family of all FDs that holds in R . Then $F = F_R$ satisfies

- (F1) $X \rightarrow X \in F$,
- (F2) $(X \rightarrow Y \in F, Y \rightarrow Z \in F) \Rightarrow (X \rightarrow Z \in F)$,
- (F3) $(X \rightarrow Y \in F, X \subseteq V, W \subseteq Y) \Rightarrow (V \rightarrow W \in F)$,
- (F4) $(X \rightarrow Y \in F, V \rightarrow W \in F) \Rightarrow (X \cup V \rightarrow Y \cup W \in F)$.

A family of FDs satisfying (F1) - (F4) is called an f -family over U .

Clearly, F_R is an f -family over U . It is known [1] that if F is an arbitrary f -family, then there is a relation R over U such that $F_R = F$.

Given a family F of FDs over U , there exists a unique minimal f -family F^+ that contains F . It can be seen that F^+ contains all FDs which can be derived from F by the rules (F1) - (F4).

A relation scheme s is a pair (U, F) , where U is a set of attributes and F is a set of FDs over U .

Let U be a nonempty finite set and $\mathcal{P}(U)$ its power set. The mapping $\mathcal{L} : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ is called a closure operation over U if it satisfies the following conditions:

- (1) $X \subseteq \mathcal{L}(X)$,
- (2) $X \subseteq Y$ implies $\mathcal{L}(X) \subseteq \mathcal{L}(Y)$,
- (3) $\mathcal{L}(\mathcal{L}(X)) = \mathcal{L}(X)$.

Remark 1.1. It is clear that, if F is an f -family, and we define $\mathcal{L}_F(X)$ as

$$\mathcal{L}_F(X) = \{a \in U : X \rightarrow \{a\} \in F\}$$

then \mathcal{L}_F is a closure operation over U . Conversely, it is known [1, 3] that if \mathcal{L} is a closure operation, then there is exactly one f -family F over U so that $\mathcal{L} = \mathcal{L}_F$, where

$$F = \{X \rightarrow Y : X, Y \subseteq U, Y \subseteq \mathcal{L}(X)\}.$$

Thus, there is a one-to-one correspondence between closure operations and f -families over U .

Let R be a relation over U and $K \subseteq U$. Then K is a key of R if $K \rightarrow U \in F_R$. K is a minimal key of R if K is a key of R and any proper subset of K is not a key of R .

Denote K_R the set of all minimal keys of R .

Let $I \subseteq \mathcal{P}(U)$, $U \in I$, and $A, B \in I \Rightarrow A \cap B \in I$. I is called a meet-semilattice over U . Let $M \subseteq \mathcal{P}(U)$. Denote $M^+ = \{\cap M' : M' \subseteq M\}$. We say that M is a generator of I if $M^+ = I$. Note that $U \in M^+$ but not in M , by convention it is the intersection of the empty collection of sets.

Denote $N = \{A \in I : A \neq \cap \{A' \in I : A \subset A'\}\}$. It can be seen that N is the unique minimal generator of I .

2 Hypergraphs and Transversals

Let U be a nonempty finite set and put $\mathcal{P}(U)$ for the family of all subsets of U . The family $\mathcal{H} = \{E_i : E_i \in \mathcal{P}(U), i = 1, 2, \dots, m\}$ is called a *hypergraph* over U if $E_i \neq \emptyset$ holds for all i (in [2] it is required that the union of E_i s is U , in this paper we do not require this).

The elements of U are called vertices, and the sets E_1, \dots, E_m the edges of the hypergraph \mathcal{H} .

A hypergraph \mathcal{H} is called *simple* if it satisfies $\forall E_i, E_j \in \mathcal{H} : E_i \subseteq E_j \Rightarrow E_i = E_j$. It can be seen that K_R is a simple hypergraph.

Let \mathcal{H} be a hypergraph over U . Then $\min(\mathcal{H})$ denotes the set of minimal edges of \mathcal{H} with respect to set inclusion, i.e., $\min(\mathcal{H}) = \{E_i \in \mathcal{H} : \nexists E_j \in \mathcal{H} : E_j \subset E_i\}$, and $\max(\mathcal{H})$ denotes the set of maximal edges of \mathcal{H} with respect to set inclusion, i.e., $\max(\mathcal{H}) = \{E_i \in \mathcal{H} : \nexists E_j \in \mathcal{H} : E_j \supset E_i\}$.

It is clear that, $\min(\mathcal{H})$ and $\max(\mathcal{H})$ are simple hypergraphs. Furthermore, $\min(\mathcal{H})$ and $\max(\mathcal{H})$ are uniquely determined by \mathcal{H} .

A set $T \subseteq U$ is called a *transversal* of \mathcal{H} (sometimes it is called *hitting set*) if it meets all edges of \mathcal{H} , i.e., $\forall E \in \mathcal{H} : T \cap E \neq \emptyset$. Denote by $\text{Trs}(\mathcal{H})$ the family of all transversals of \mathcal{H} . A transversal T of \mathcal{H} is called *minimal* if no proper subset T' of T is a transversal.

The family of all minimal transversals of \mathcal{H} called the *transversal hypergraph* of \mathcal{H} , and denoted by $\text{Tr}(\mathcal{H})$. Clearly, $\text{Tr}(\mathcal{H})$ is a simple hypergraph.

Proposition 2.1 ([2]). *Let \mathcal{H} and \mathcal{G} two simple hypergraphs over U . Then*

- (1) $\mathcal{H} = \text{Tr}(\mathcal{G})$ if and only if $\mathcal{G} = \text{Tr}(\mathcal{H})$,
- (2) $\text{Tr}(\mathcal{H}) = \text{Tr}(\mathcal{G})$ if and only if $\mathcal{H} = \mathcal{G}$,
- (3) $\text{Tr}(\text{Tr}(\mathcal{H})) = \mathcal{H}$.

By the definition of minimal transversal, the following proposition is obvious

Proposition 2.2. *Let \mathcal{H} be a hypergraph over U . Then*

$$\text{Tr}(\mathcal{H}) = \text{Tr}(\min(\mathcal{H})).$$

The following algorithm finds the family of all minimal transversals of a given hypergraph (by induction).

Algorithm 2.3 ([5]).

Input: let $\mathcal{H} = \{E_1, \dots, E_m\}$ be a hypergraph over U .

Output: $\text{Tr}(\mathcal{H})$.

Method:

Step 0. We set $L_1 := \{\{a\} : a \in E_1\}$. It is obvious that $L_1 = \text{Tr}(\{E_1\})$.

Step $q+1$. ($q < m$) Assume that

$$L_q = S_q \cup \{B_1, \dots, B_{t_q}\},$$

where $B_i \cap E_{q+1} = \emptyset, i = 1, \dots, t_q$ and $S_q = \{A \in L_q : A \cap E_{q+1} \neq \emptyset\}$.

For each i ($i = 1, \dots, t_q$) constructs the set $\{B_i \cup \{b\} : b \in E_{q+1}\}$. Denote them by $A_1^i, \dots, A_{r_i}^i$ ($i = 1, \dots, t_q$). Let

$$L_{q+1} = S_q \cup \{A_p^i : A \in S_q \Rightarrow A \not\subset A_p^i, 1 \leq i \leq t_q, 1 \leq p \leq r_i\}.$$

Theorem 2.4 ([5]). *For every q ($1 \leq q \leq m$) $L_q = Tr(\{E_1, \dots, E_q\})$, i.e., $L_m = Tr(\mathcal{H})$.*

It can be seen that the determination of $Tr(\mathcal{H})$ based on our algorithm does not depend on the order of E_1, \dots, E_m .

Remark 2.5. Denote $L_q = S_q \cup \{B_1, \dots, B_{t_q}\}$, and l_q ($1 \leq q \leq m-1$) be the number of elements of L_q . It can be seen that the worst-case time complexity of our algorithm is

$$\mathcal{O}(|U|^2 \sum_{q=0}^{m-1} t_q u_q),$$

where $l_0 = t_0 = 1$ and

$$u_q = \begin{cases} l_q - t_q, & \text{if } l_q > t_q; \\ 1, & \text{if } l_q = t_q. \end{cases}$$

Clearly, in each step of our algorithm L_q is a simple hypergraph. It is known that the size of arbitrary simple hypergraph over U cannot be greater than $C_n^{\lfloor n/2 \rfloor}$, where $n = |U|$. $C_n^{\lfloor n/2 \rfloor}$ is asymptotically equal to $2^{n+1/2}/(\pi \cdot n)^{1/2}$. From this, the worst-case time complexity of our algorithm cannot be more than exponential in the number of attributes. In cases for which $l_q \leq l_m$ ($q = 1, \dots, m-1$), it is easy to see that the time complexity of our algorithm is not greater than $\mathcal{O}(|U|^2 |\mathcal{H}| |Tr(\mathcal{H})|^2)$. Thus, in these cases this algorithm finds $Tr(\mathcal{H})$ in polynomial time in $|U|$, $|\mathcal{H}|$ and $|Tr(\mathcal{H})|$. Obviously, if the number of elements of \mathcal{H} is small, then this algorithm is very effective. It only requires polynomial time in $|R|$.

The following proposition is obvious

Proposition 2.6 ([5]). *The time complexity of finding $Tr(\mathcal{H})$ of a given hypergraph \mathcal{H} is (in general) exponential in the number of elements of U .*

Proposition 2.6 is still true for a simple hypergraph.

3 Dense Families

Let $\mathcal{D} \subseteq \mathcal{P}(U)$ be a family of subsets of a U . We define a set $F_{\mathcal{D}}$ over \mathcal{D} as follows

$$F_{\mathcal{D}} = \{X \rightarrow Y : (\forall A \in \mathcal{D}) X \subseteq A \Rightarrow Y \subseteq A\}.$$

Proposition 3.1 ([7]). *If \mathcal{D} is a family of subsets of a finite set U , then $F_{\mathcal{D}}$ is an f -family over U .*

The notion of dense family of a database relation is defined in [7], as follows:

Let R be a relation over U . We say that a family $\mathcal{D} \subseteq \mathcal{P}(U)$ of attribute sets is R -dense (or dense in R) if $F_R = F_{\mathcal{D}}$.

The following proposition guarantees the existence of at least one dense family. In the sequel we denote \mathcal{L}_{F_R} simply by \mathcal{L}_R .

Proposition 3.2 ([7]). *The family \mathcal{L}_R is R -dense.*

Proposition 3.3 ([7]). *If \mathcal{D} is R -dense, then $\mathcal{D} \subseteq \mathcal{L}_R$.*

Note that by Proposition 3.2 and Proposition 3.3, \mathcal{L}_R is the greatest R -dense family.

For any $A \subseteq U$, we denote by \bar{A} the complement of A with respect to the set U , that is, $\bar{A} = \{a \in U : a \notin A\}$.

Theorem 3.4 ([7]). *Let R be a relation over U . If $\mathcal{D} \subseteq \mathcal{P}(U)$ is R -dense, then the following conditions hold*

(1) *K is a key of R if and only if it contains an element from each set in $\{\bar{A} : A \in \mathcal{D}, A \neq U\}$.*

(2) *K is a minimal key of R if and only if it is minimal with respect to the property of containing an element from each set in $\{\bar{A} : A \in \mathcal{D}, A \neq U\}$.*

Let U be a finite set and $\mathcal{P}(U)$ its power set. For every family $\mathcal{D} \subseteq \mathcal{P}(U)$, the complement family of \mathcal{D} is the family $\bar{\mathcal{D}} = \{\bar{A} : A \in \mathcal{D}\}$ over U .

Let $R = \{h_1, \dots, h_m\}$ be a relation over U , and E_R the equality set of R , i.e.,

$$E_R = \{E_{ij} : 1 \leq i < j \leq m\}$$

where $E_{ij} = \{a \in U : h_i(a) = h_j(a)\}$.

Proposition 3.5. *The equality set E_R is R -dense.*

Proof. Assume that $X \rightarrow Y \in F_R$. Let $E_{ij} \in E_R$ such that $X \subseteq E_{ij}$. This means that $h_i(X) = h_j(X)$. From this, and according to the definition of FDs, we have $h_i(Y) = h_j(Y)$. Thus, $Y \subseteq E_{ij}$. By the definition of F_{E_R} , that is,

$$F_{E_R} = \{X \rightarrow Y : (\forall E_{ij} \in E_R) X \subseteq E_{ij} \Rightarrow Y \subseteq E_{ij}\},$$

we obtain $X \rightarrow Y \in F_{E_R}$.

Conversely, let $X \rightarrow Y \in F_{E_R}$. Suppose that there are $h_i, h_j \in R$ such that $h_i(X) = h_j(X)$, $1 \leq i < j \leq m$. Which means that $X \subseteq E_{ij}$. By $X \rightarrow Y \in F_{E_R}$, $Y \subseteq E_{ij}$. Hence, we also obtain $h_i(Y) = h_j(Y)$. Consequently, $X \rightarrow Y \in F_R$.

The proposition is proved. \square

It is easy to see that the dense family E_R has at most $\frac{m(m-1)}{2}$ elements. By Proposition 3.3, we also have $E_R \subseteq \mathcal{L}_R$.

Theorem 3.6. *Let R be a relation over U . Then*

$$K_R = \text{Tr}(\min(\overline{E_R})).$$

Proof. By the definition of relation R , we have $U \notin E_R$. From this, Proposition 2.2, Proposition 3.5 and Theorem 3.4, the theorem is obvious.

The proof is complete. \square

Let $R = \{h_1, \dots, h_m\}$ be a relation over U , and N_R the nonequality set of R , i.e.,

$$N_R = \{N_{ij} : 1 \leq i < j \leq m\}$$

where $N_{ij} = \{a \in U : h_i(a) \neq h_j(a)\}$.

Note that, because R is a relation, $\emptyset \notin N_R$ and $U \notin E_R$. Moreover, $N_R = \overline{E_R}$. From this, and Theorem 3.6, the following corollary is immediate

Corollary 3.7. *Let R be a relation over U . Then*

$$K_R = \text{Tr}(\min(N_R)).$$

Corollary 3.7 was shown in [5].

Proposition 3.8. *If \mathcal{D} is R -dense, then*

$$\min(\overline{\mathcal{D}} - \{\emptyset\}) = \overline{\max(E_R)}.$$

Proof. According to Theorem 3.6, we have $K_R = \text{Tr}(\overline{E_R})$. By Proposition 2.2, it is clear that

$$K_R = \text{Tr}(\overline{\max(E_R)}). \quad (1)$$

Because \mathcal{D} is R -dense, and by Theorem 3.4, we have $K_R = \text{Tr}(\overline{\mathcal{D}} - \{\emptyset\})$. Furthermore, we have

$$\text{Tr}(\overline{\mathcal{D}} - \{\emptyset\}) = \text{Tr}(\min(\overline{\mathcal{D}} - \{\emptyset\})).$$

Hence

$$K_R = \text{Tr}(\min(\overline{\mathcal{D}} - \{\emptyset\})). \quad (2)$$

From (1) and (2), we give

$$\text{Tr}(\min(\overline{\mathcal{D}} - \{\emptyset\})) = \text{Tr}(\overline{\max(E_R)}).$$

By $\min(\overline{\mathcal{D}} - \{\emptyset\})$ and $\overline{\max(E_R)}$ are simple hypergraphs, thus according to Proposition 2.1 we have

$$\min(\overline{\mathcal{D}} - \{\emptyset\}) = \overline{\max(E_R)}.$$

The proposition is proved. \square

From Proposition 3.8, the following corollary is clear

Corollary 3.9. *If \mathcal{D} is R -dense, then*

$$\min(\overline{\mathcal{D}} - \{\emptyset\}) = \min(N_R).$$

Now we give a necessary and sufficient condition for an arbitrary family \mathcal{D} is R -dense.

Theorem 3.10. *Let R be a relation, $\mathcal{D} \subseteq \mathcal{P}(U)$ a family of subsets of a U . Then \mathcal{D} is R -dense iff for every $X \subseteq U$*

$$\mathcal{L}_R(X) = \begin{cases} \bigcap_{X \subseteq A} A & \text{if } \exists A \in \mathcal{D} : X \subseteq A, \\ U & \text{otherwise,} \end{cases}$$

where $\mathcal{L}_R(X) = \{a \in U : X \rightarrow \{a\} \in F_R\}$.

Proof. First we prove that in an arbitrary family $\mathcal{D} \subseteq \mathcal{P}(U)$ for all $X \subseteq U$

$$\mathcal{L}_{F_{\mathcal{D}}}(X) = \begin{cases} \bigcap_{X \subseteq A} A & \text{if } \exists A \in \mathcal{D} : X \subseteq A, \\ U & \text{otherwise.} \end{cases}$$

Suppose that X is a set such that there is no $A \in \mathcal{D}$ with $X \subseteq A$. By the definition of $F_{\mathcal{D}}$, it is easy to see that $X \rightarrow U \in F_{\mathcal{D}}$. Hence, $\mathcal{L}_{F_{\mathcal{D}}}(X) = U$.

Since $\emptyset \subseteq \bigcap_{A \in \mathcal{D}} A \subseteq A$, according to the definition of $F_{\mathcal{D}}$ and $\mathcal{L}_{F_{\mathcal{D}}}$ we obtain

$$\mathcal{L}_{F_{\mathcal{D}}}(\emptyset) = \bigcap_{A \in \mathcal{D}} A.$$

If $X \neq \emptyset$ and there is an $A \in \mathcal{D}$ such that $X \subseteq A$ then we set

$$\mathcal{G} = \{A : X \subseteq A, A \in \mathcal{D}\},$$

$$B = \bigcap_{A \in \mathcal{G}} A.$$

It is easy to see that $X \subseteq B$ holds. If $\mathcal{G} = \mathcal{D}$ or $\mathcal{G} \neq \mathcal{D}$, then we also obtain $X \rightarrow B \in F_{\mathcal{D}}$.

By the definition of $\mathcal{L}_{F_{\mathcal{D}}}$, we have $B \subseteq \mathcal{L}_{F_{\mathcal{D}}}(X)$. Using $X \subseteq B \subseteq \mathcal{L}_{F_{\mathcal{D}}}(X)$, we obtain $B \rightarrow \mathcal{L}_{F_{\mathcal{D}}}(X) \in F_{\mathcal{D}}$.

Now we suppose that b is an attribute such that $b \notin B$. Then, there is $A \in \mathcal{G}$ so that $b \notin A$. Hence, by the definition of $F_{\mathcal{D}}$ we have $B \rightarrow B \cup \{b\} \notin F_{\mathcal{D}}$. Consequently,

$$\mathcal{L}_{F_{\mathcal{D}}}(X) = \bigcap_{A \in \mathcal{D}} (A).$$

By Remark 1.1 it is easy to see that $F_R = F_{\mathcal{D}}$ holds iff $\mathcal{L}_R = \mathcal{L}_{F_{\mathcal{D}}}$ does. The Theorem is proved. \square

From Theorem 3.10 and Proposition 3.5, the following proposition is obvious

Proposition 3.11. *Let $R = \{h_1, \dots, h_m\}$ be a relation over $U = \{a_1, \dots, a_n\}$. Then*

- (1) *If \mathcal{D} is R -dense, then $\mathcal{D} \cup \{U\}$ also is R -dense, and thus $E_R \cup \{U\}$ is R -dense.*
- (2) *If $m = 1$ or $F_R = \{\{a_1\} \rightarrow U, \dots, \{a_n\} \rightarrow U\}$, then families $\mathcal{D}_1 = \emptyset$, $\mathcal{D}_2 = \{\emptyset\}$ and $\mathcal{D}_3 = \{U\}$ are R -densities.*

4 Finding the set of all minimal keys of a relation

In this section, we give the following algorithm finding all minimal keys of a given relation R . Remember that this problem is inherently exponential in the size of R [4].

Algorithm 4.1.

Input: a relation $R = \{h_1, \dots, h_m\}$ over U .

Output: K_R .

Method:

Step 1. Construct the equality set

$$E_R = \{E_{ij} : 1 \leq i < j \leq m\}$$

where $E_{ij} = \{a \in U : h_i(a) = h_j(a)\}$.

Step 2. Compute the complement of E_R as follows

$$\overline{E_R} = \{\overline{E_{ij}} : E_{ij} \in E_R\}.$$

Denote elements of $\overline{E_R}$ by N_1, \dots, N_k

Step 3. From $\overline{E_R}$ compute the family $\min(\overline{E_R}) = \{N_i \in \overline{E_R} : \nexists N_j \in \overline{E_R} : N_i \subseteq N_j\}$.

Step 4. By Algorithm 2.3 we construct the set $Tr(\min(\overline{E_R}))$.

Based on Proposition 2.2, Algorithm 2.3 and Theorem 3.6, we have $K_R = Tr(\min(\overline{E_R}))$. It can be seen that the time complexity of this algorithm is the time complexity of Algorithm 2.3. In many cases this algorithm is very effective (see Remark 2.5).

It can be seen that, if the number of elements of the equality set E_R is constant, i.e. $|E_R| \leq k$ for some constant k , then the time complexity of finding K_R of a given relation R is polynomial time [9].

The following example shows that for a given relation R , Algorithm 4.1 can be applied to find all minimal keys of a given relation R .

Example 4.2. Let us consider the relation R over $U = \{a, b, c, d\}$ as follows

	a	b	c	d
	0	0	0	0
	0	0	0	1
$R = 2$	0	0	0	0
	3	3	0	0
	4	0	4	4
	5	5	5	0

It can be seen that the equality set E_R is the following

$$E_R = \{\emptyset, \{b\}, \{c\}, \{d\}, \{b, c\}, \{c, d\}, \{a, b, c\}, \{b, c, d\}\}.$$

Hence

$$\overline{E_R} = \{\{a\}, \{d\}, \{a, d\}, \{a, b\}, \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, U\},$$

$$\min(\overline{E_R}) = \{\{a\}, \{d\}\}.$$

From this, we obtain

$$K_R = \{\{a, d\}\}.$$

5 Finding the cover of a relation

From Proposition 3.5 and Theorem 3.10 we have an application, which is the following algorithm finding a cover of FDs of a given relation R . Recall that this problem is inherently exponential in the size of R [6].

Algorithm 5.1.

Input: a relation $R = \{h_1, \dots, h_m\}$ over U .

Output: F_R .

Method:

Step 1. Construct the equality set

$$E_R = \{E_{ij} : 1 \leq i < j \leq m\}$$

where $E_{ij} = \{a \in U : h_i(a) = h_j(a)\}$.

Step 2. Compute the family $E_R^+ = \{\cap \mathcal{A} : \mathcal{A} \subseteq E_R\}$. Denote the elements of E_R^+ by X_1, \dots, X_t .

Step 3. Construct set of FDs as follows

$$F = \{K_1 \rightarrow X_1 : K_1 \in \text{Key}(X_1)\} \cup \dots \cup \{K_t \rightarrow X_t : K_t \in \text{Key}(X_t)\}$$

where $\text{Key}(X_i)$ is a set of all minimal keys of $\Pi_{X_i}(R)$ (the projection of R onto the attributes set X_i).

Obviously, $F = F_R$. Note that $\mathcal{L}_R = E_R^+$. It is easy to see that the time complexity of this algorithm is exponential in the number of attributes.

The following example shows that for a given relation R , Algorithm 5.1 can be applied to find a cover of a given relation R .

Example 5.2. R is the following relation over $U = \{a, b, c, d\}$

$$R = \begin{array}{ccc} & a & b & c \\ \begin{array}{c} 0 \\ 0 \\ 1 \end{array} & \begin{array}{c} 0 \\ 1 \\ 1 \end{array} & \begin{array}{c} 0 \\ 0 \\ 0 \end{array} \end{array}$$

It can be seen that the equality set E_R is the following

$$E_R = \{\{c\}, \{a, c\}, \{b, c\}\}.$$

Therefore

$$E_R^+ = \{\{c\}, \{a, c\}, \{b, c\}, U\}.$$

From this, we have

$$F = \{\{a\} \rightarrow \{c\}, \{b\} \rightarrow \{c\}, \{a, b\} \rightarrow \{c\}\}.$$

It is obvious that $F = F_R$.

References

- [1] Armstrong W. W., *Dependency structure of database relationship*, Information Processing 74, North-Holland Pub. Co. , (1974) 580-583.
- [2] Berge C., *Hypergraphs: combinatorics of finite sets*, North - Holland, Amsterdam (1989).
- [3] Demetrocic J., *On the equivalence of candidate keys with Sperner systems*, Acta Cybernetica 4, (1979), 247-252.
- [4] Demetrovics J., Thi V.D., *Keys, antikeys and prime attributes*, Annales Univ. Sci. Budapest Sect. Comp. 8, (1987), 35-52.
- [5] Demetrovics J., Thi V. D., *Describing candidate keys by hypergraphs*, Computers and Artificial Intelligence 18, 2 (1999), 191-207.
- [6] Gottlob G., Libkin L., *Investigations on Armstrong relations, dependency inference, and excluded functional dependencies*, Acta Cybernetica Hungary 9, 4 (1990), 385-402.
- [7] Järvinen J., *Dense families and key functions of database relation instances*, in: Freivalds R. (ed.), Fundamentals of Computation Theory, Proceedings of the 13th International Symposium, Lecture Notes in Computer Science 2138 (Springer-Verlag, Heidelberg, 2001), 184-192.
- [8] Thi V. D., *Minimal keys and antikeys*, Acta Cybernetica 7 (1986), 361-371.
- [9] Thi V. D., Son N. H., *Some problems related to keys and the Boyce-Codd normal form*, Acta Cybernetica 16, 3 (2004), 473-483.

Received December, 2004

CONTENTS

<i>Balázs Imreh and Masami Ito</i> : On regular languages determined by nonde- terministic directable automata	1
<i>Alexander Meduna and Jiří Techet</i> : Generation of Sentences with Their Parses: the Case of Propagating Scattered Context Grammars	11
<i>Saeed Salehi</i> : Varieties of Tree Languages Definable by Syntactic Monoids . .	21
<i>Ludwig Staiger</i> : Topologies for the Set of Disjunctive ω -words	43
<i>Bianca Truthe</i> : On the Finiteness of Picture Languages of Synchronous De- terministic Chain Code Picture Systems	53
<i>Zhilin Wu</i> : Quasi-star-free Languages on Infinite Words	75
<i>József Dombi and Nándor Vincze</i> : The lexicographic decision function	95
<i>Tamás Vinkó</i> : Minimal inter-particle distance in atom clusters	107
<i>Balázs Ugron, Szabolcs Hajdara, and László Kozma</i> : Synthesis of the syn- chronization of general pipeline systems	123
<i>Sven Hartmann, Sebastian Link, and Klaus-Dieter Schewe</i> : Functional De- pendencies over XML Documents with DTDs	153
<i>Vu Duc Thi and Nguyen Hoang Son</i> : Some Results Related to Dense Families of Database Relations	173

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János